SONiX Technology Co., Ltd.

# SN8F5702 Series
# Datasheet

8051-based Microcontroller

SN8F5702
SN8F570200
SN8F570202
SN8F570210
SN8F570211
SN8F570212
SN8F570213

# 1 Device Overview

## 1.1 Features

- **Enhanced 8051 microcontroller** with reduced instruction cycle time (up to 12 times 80C51)
- Up to 8 MHz flexible CPU frequency
- Internal 32 MHz Clock Generator (IHRC)
- **4 KB non-volatile flash memory (IROM)** with in-system program support
- **256 bytes internal RAM (IRAM)**
- **13 interrupt sources with priority levels control and unique interrupt vectors**
- 12 internal interrupts
- 1 external interrupts: INT0
- 1 set of DPTR
- 2 set 8/16-bit timers with 4 operation modes
- 1 set 16-bit timers with 4 comparison output (PWM) and capture channels
- **1 set 16-bit PWM generators**:
- each PWM generator has 4 output channels with inverters and dead-band control
- **12-bit SAR ADC** with 10 external and 1 internal channels, and 4 internal reference voltages
- **SPI, UART, I2C** interface with SMBus Support
- **On-Chip Debug Support**:
  Single-wire debug interface
  2 hardware breakpoints
  Unlimited software breakpoints
  ROM data security/protection
- Watchdog and programmable external reset
- 1.8V low voltage detectors
- Wide supply voltage (1.8 V – 5.5 V) and temperature (-40 °C to 85 °C) range

## 1.2 Applications

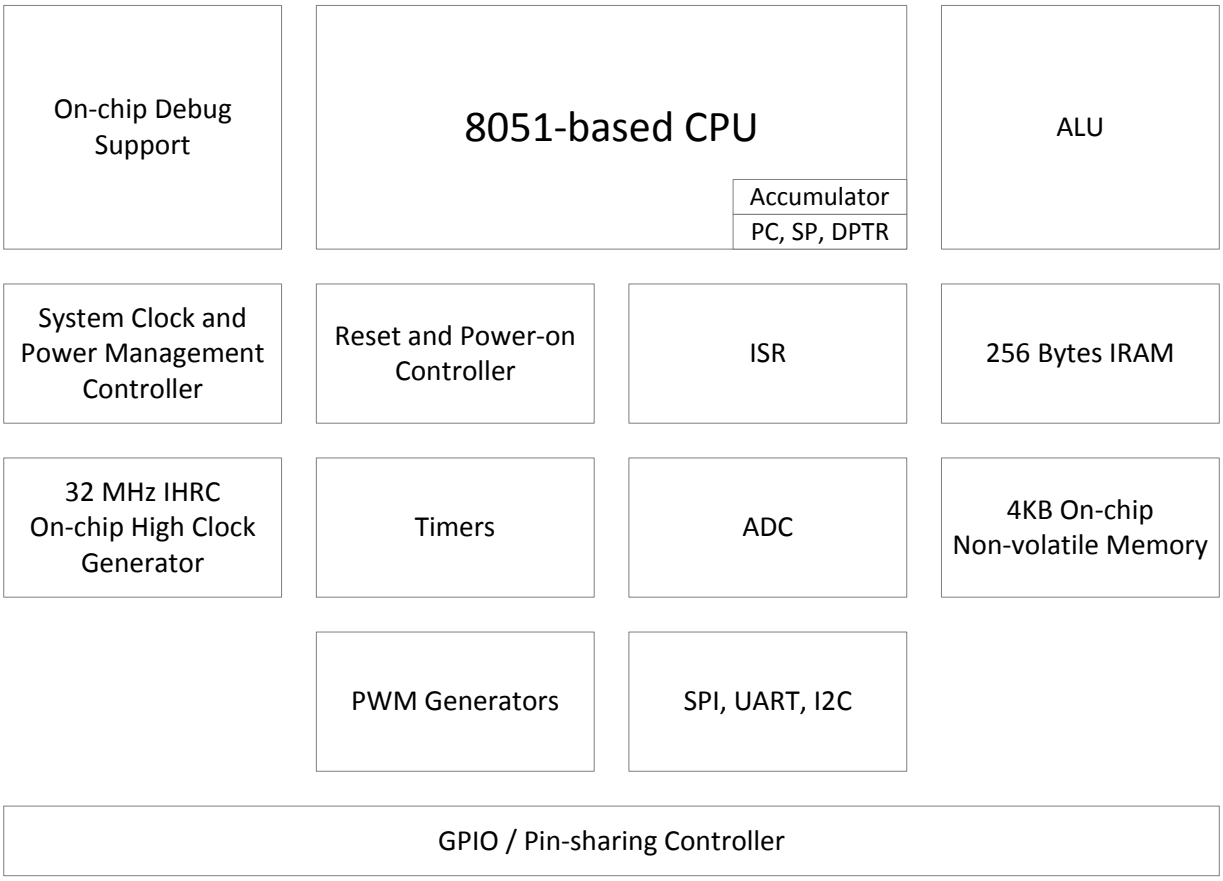
- Brushless DC motor
- Home automation
- Household
- Other

## 1.3 Features Selection Table

| | I/O | PWM Channels | I2C | SPI | UART | ADC ext. Channels | OPA | CMP | Ext. INT | Package Types |
|---|---|---|---|---|---|---|---|---|---|---|
| SN8F5702 | 18 | 8 | V | V | V | 10 | - | - | 1 | DIP20,SOP20 TSSOP20, QFN20 |
| SN8F570212 | 14 | 6 | V | V | TX*(1) | 8 | - | - | 1 | SOP16,TSSOP16 |
| SN8F570210 | 12 | 5 | - | - | V | 6 | - | - | 1 | SOP14 |
| SN8F570211 | 12 | 5 | V | - | TX*(1) | 6 | - | - | 1 | SOP14 |
| SN8F570213 | 12 | 5 | - | - | V | 6 | - | - | 1 | SOP14 |
| SN8F570200 | 8 | 3 | - | - | TX*(1) | 5 | - | - | 1 | MSOP10 |
| SN8F570202 | 6 | 3 | - | - | TX*(1) | 4 | - | - | 1 | SOP8 |

*(1) Only support UART TX mode.

## 1.4 Block Diagram



## 2 Table of Contents

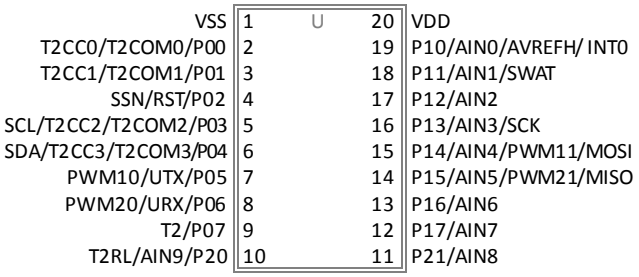
# 3 Revision History

| Revision | Date | Description |
|---|---|---|
| 1.0 | Sep. 2015 | First issue |
| 1.1 | Oct. 2015 | 1. Modify timer section and electrical characteristic section.<br>2. Modify SN8F57023/SN8F57024 pin assignment.<br>3. Add program memory security section, special function register section and noise filter section.<br>4. Modify minimum requirement in debug interface section.<br>5. Update electrical characteristics |
| 1.2 | Nov. 2015 | 1. Update package type |
| 1.3 | Nov. 2015 | 1. SN8F57021 was renamed SN8F570210.<br>2. SN8F57022 was renamed SN8F570200.<br>3. SN8F57023 was renamed SN8F570211.<br>4. SN8F57024 was renamed SN8F570212. |
| 1.4 | Nov. 2015 | 1. Modify SN8F570200 pin assignment |
| 1.5 | Dec. 2015 | 1. Modify electrical characteristic in IHRC section.<br>2. Add power saving description in UART/SPI/I2C section. |
| 1.6 | Apr. 2016 | 1. Add Timer 2 capture function waveform to illustrate operation.<br>2. Special Function Registers adds Register Declaration section.<br>3. Add Appendix: Reference Document chapter.<br>4. Add ROM Programming Pin chapter.<br>5. Add QFN20 and SOP14 package type. |
| 1.7 | Aug. 2016 | 1. I2C example modify.<br>2. Modify Power Management section and In-System Program section.<br>3. Modify PW1M & PW1YH/L registers description.<br>4. Add SOP8 package type.<br>5. Add ADC internal reference range. |
| 1.8 | Nov. 2016 | 1. Modify feature table and I2C status code.<br>2. Add UART Baud Rate Table, WDT description in watchdog reset section and QFN16 package type. |
| 1.9 | Dec. 2016 | 1. Modify electrical characteristic section. |
| 2.0 | Sep. 2017 | 1. Add pin circuit diagrams section.<br>2. Add package information. |
| 2.1 | Nov. 2017 | 1. Modify LVD related content. |
| 2.2 | Dec. 2017 | 1. Add design note description. |
| 2.3 | Jul. 2018 | 1. Modify QFN16 3x3 dimension. |

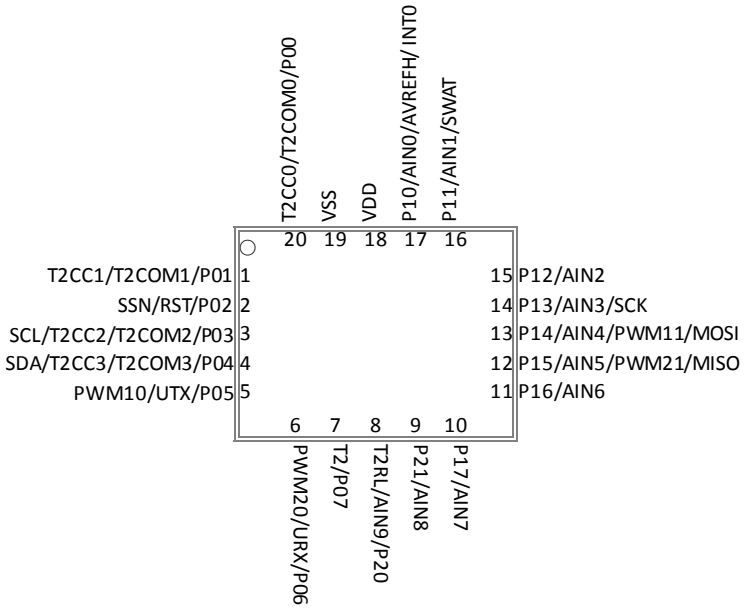
| | | |
|---|---|---|
| | | 2. Modify P0OC register in UART section.<br>3. Repair an error, omission, etc.<br>4. Add Pin Characteristic section.<br>5. Modify Internal & External RAM section description.<br>6. Modify Program Memory section description.<br>7. Modify Configuration of Reset and Power-on Controller section description.<br>8. Modify System clock section description.<br>9. Add High Speed Clock and Real time clock section.<br>10. Add System clock timing section.<br>11. Add System Operating Mode chapter.<br>12. Modify Interrupt Priority section description.<br>13. Interrupt chapter adds example section.<br>14. Modify UART chapter description and baud rate table.<br>15. I2C chapter adds protocol description diagram and modifies the clock rate table.<br>16. Debug Interface chapter was renamed Development Environment chapter. Modify Development Environment chapter description. Add Development Tool section.<br>17. Add SN5702 Starter-kit chapter.<br>18. Modify ROM Programming Pin chapter description. Add MP5 Hardware Connecting, SN-Link ISP Programming and SN-Link ISP Programming Pin Mapping sections.<br>19. Update Device Nomenclature section. |
| 2.4 | Mar. 2019 | 1. Repair an error, omission, etc.<br>2. Modify Pin Circuit Diagrams section.<br>3. Modify T2 Comparison Output section.<br>4. Modify Starter-Kit section.<br>5. Modify Package Information section.<br>6. Modify SPI operation section description.<br>7. Modify Timer0/ Timer1 section description.<br>8. Remove SN8F570212JG package type. |

# 4 Pin Assignments

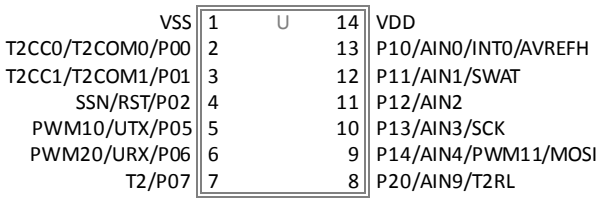
## 4.1 SN8F5702P/S/T (DIP20/SOP20/TSSOP20)



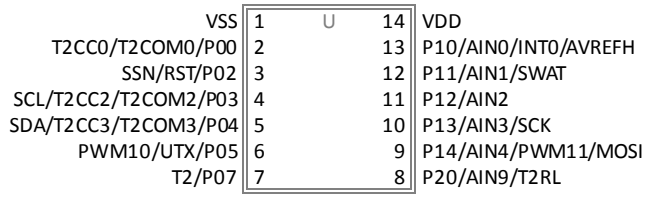

## 4.2 SN8F5702J (QFN20)



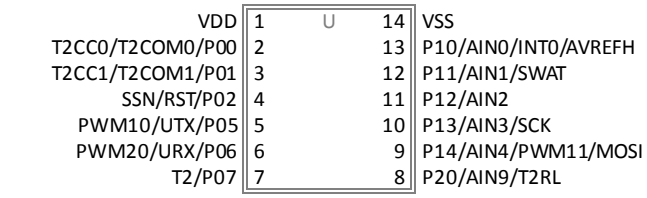

## 4.3 SN8F570210S (SOP14)



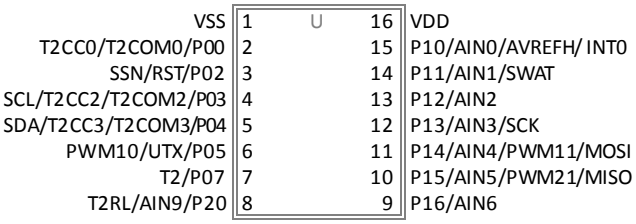
## 4.4 SN8F570211S (SOP14)



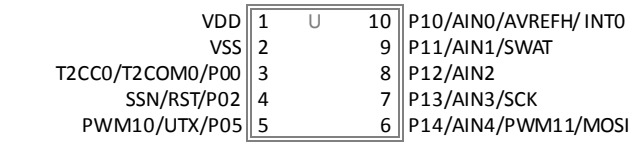

## 4.5 SN8F570213S (SOP14)



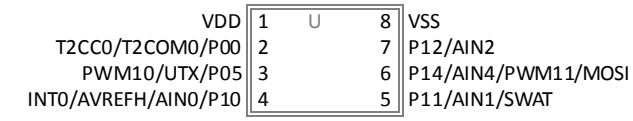

## 4.6 SN8F570212S/T (SOP16/TSSOP16)




## 4.7 SN8F570200A (MSOP10)




## 4.8 SN8F570202S (SOP8)



## 4.9 Pin Descriptions

**Power Pins**

| Pin Name | Type | Description |
|---|---|---|
| VDD | Power | Power supply |
| VSS | Power | Ground (0 V) |

**Port 0**

| Pin Name | Type | Description |
|---|---|---|
| P0.0 | Digital I/O | GPIO: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up. |
| T2COM0 | Digital Output | Timer 2: compare 0 output. |
| T2CC0 | Digital Input | Timer 2: capture 0 input. |
| P0.1 | Digital I/O | GPIO: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up. |
| T2COM1 | Digital Output | Timer 2: compare 1 output. |
| T2CC1 | Digital Input | Timer 2: capture 1 input. |
| P0.2 | Digital I/O | GPIO: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up. |
| Reset | Digital Input | System reset (active low). |
| SSN | Digital Input | SPI: salve selection pin (slave mode). |
| P0.3 | Digital I/O | GPIO: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up. |
| T2COM2 | Digital Output | Timer 2: compare 2 output. |
| T2CC2 | Digital Input | Timer 2: capture 2 input. |
| SCL | Digital I/O | I2C: clock output (master) clock input (slave). |
| P0.4 | Digital I/O | GPIO: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up. |
| T2COM3 | Digital Output | Timer 2: compare 3 output. |
| T2CC3 | Digital Input | Timer 2: capture 3 input. |
| SDA | Digital I/O | I2C: data pin. |
| P0.5 | Digital I/O | GPIO: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up. |
| UTX | Digital Output | UART: transmission pin. |
| PWM10 | Digital Output | PWM: programmable PWM output. |
| P0.6 | Digital I/O | GPIO: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up. |
| URX | Digital Input | UART: reception pin. |
| PWM20 | Digital Output | PWM: programmable PWM output. |

| | | |
|---|---|---|
| P0.7 | Digital I/O | GPIO: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up. |
| T2 | Digital Input | Timer 2: event counter input. |

**Port 1**

| Pin Name | Type | Description |
|---|---|---|
| P1.0 | Digital I/O | Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up. |
| AIN0 | Analog Input | ADC: input channel 0. |
| INT0 | Digital Input | INT0: external interrupt 0. |
| AVREFH | Analog Input | ADC: external reference voltage. |
| P1.1 | Digital I/O | Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up. |
| AIN1 | Analog Input | ADC: input channel 1. |
| SWAT | Digital I/O | Debug interface. |
| P1.2 | Digital I/O | Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up. |
| AIN2 | Analog Input | ADC: input channel 2. |
| P1.3 | Digital I/O | Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up. |
| AIN3 | Analog Input | ADC: input channel 3. |
| SCK | Digital I/O | SPI: clock output (master) clock input (slave). |
| P1.4 | Digital I/O | Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up. |
| AIN4 | Analog Input | ADC: input channel 4. |
| MOSI | Digital I/O | SPI: transmission pin (master) reception pin (slave). |
| PWM11 | Digital Output | PWM: programmable PWM output. |
| P1.5 | Digital I/O | Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up. |
| AIN5 | Analog Input | ADC: input channel 5. |
| MISO | Digital I/O | SPI: reception pin (master) transmission pin (slave). |
| PWM21 | Digital Output | PWM: programmable PWM output. |
| P1.6 | Digital I/O | Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up. |
| AIN6 | Analog Input | ADC: input channel 6. |
| P1.7 | Digital I/O | Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up. |
| AIN7 | Analog Input | ADC: input channel 7. |

**Port 2**

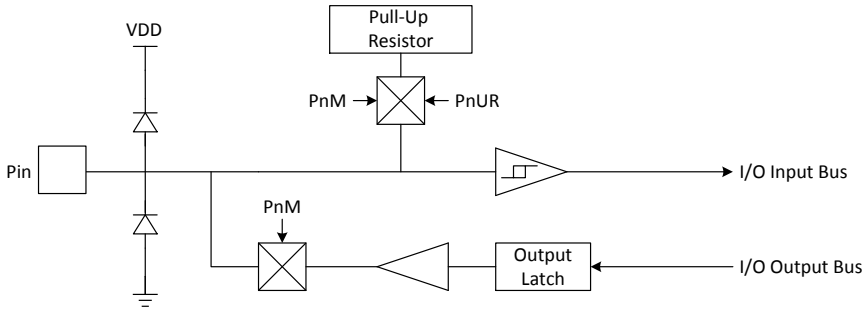
| Pin Name | Type | Description |
|---|---|---|
| P2.0 | Digital I/O | GPIO: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. |
| AIN9 | Analog Input | ADC: input channel 9. |
| T2RL | Digital Input | Timer 2: reload trigger input. |
| P2.1 | Digital I/O | GPIO: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. |
| AIN8 | Analog Input | ADC: input channel 8. |

## 4.10 Pin Characteristic

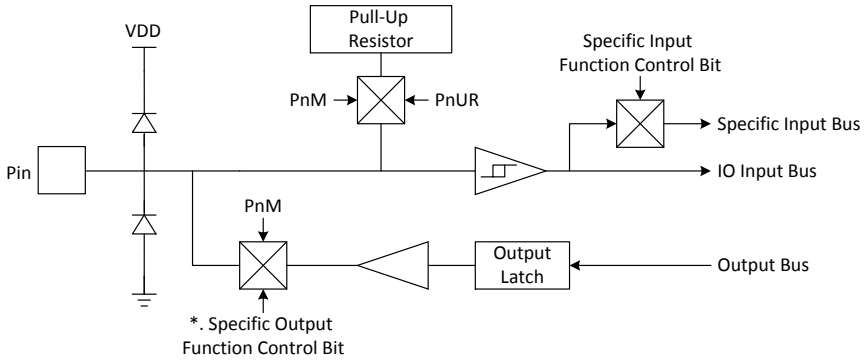
| Port | Open-Drain | Sink Current 100mA VSS+1.5V | Sink Current 20mA VSS+0.5V | External Interrupt | Wakeup (Level change) | Shared Pin |
|---|---|---|---|---|---|---|
| P0.0 | | V | - | - | V | T2CC0/T2COM0 |
| P0.1 | | V | - | - | V | T2CC1/T2COM1 |
| P0.2 | - | V | - | - | V | SSN/RST |
| P0.3 | - | V | - | - | V | SCL/T2CC2/T2COM2 |
| P0.4 | - | V | - | - | V | SDA/T2CC3/T2COM3 |
| P0.5 | V | V | - | - | V | PWM10/UTX |
| P0.6 | V | V | - | - | V | PWM20/URX |
| P0.7 | - | - | V | - | V | T2 |
| P1.0 | - | - | V | V | V | AIN0/AVREFH/INT0 |
| P1.1 | - | - | V | - | V | AIN1/SWAT |
| P1.2 | - | - | V | - | V | AIN2 |
| P1.3 | V | - | V | - | V | AIN3/SCK |
| P1.4 | V | - | V | - | V | AIN4/PWM11/MOSI |
| P1.5 | V | - | V | - | V | AIN5/PWM21/MISO |
| P1.6 | - | - | V | - | V | AIN6 |
| P1.7 | - | - | V | - | V | AIN7 |
| P2.0 | - | - | V | - | - | T2RL/AIN9 |
| P2.1 | - | - | V | - | - | AIN8 |

## 4.11 Pin Circuit Diagrams

Normal Bi-direction I/O Pin.

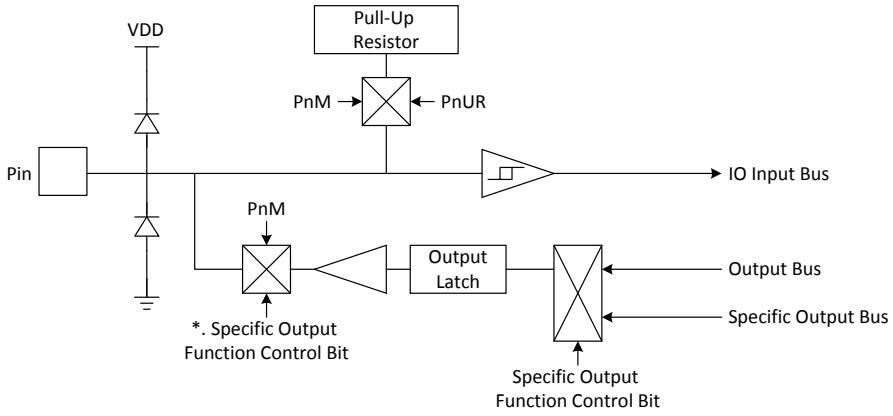



Bi-direction I/O Pin Shared with Specific Digital Input Function, e.g. INT0, Event counter, SIO, UART.



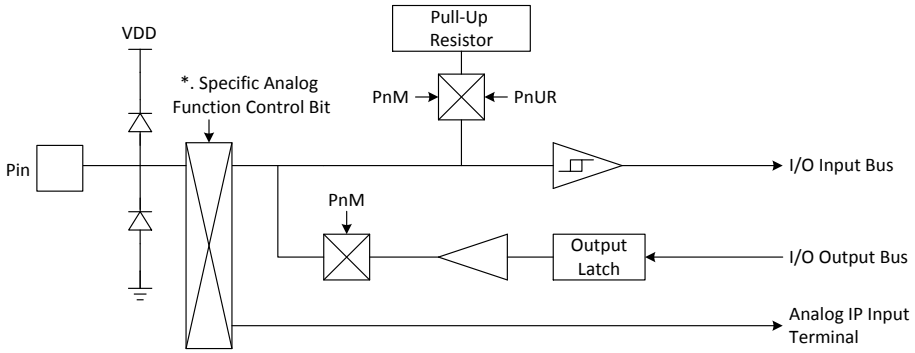

***. Some specific functions switch I/O direction directly, not through PnM register.**

Bi-direction I/O Pin Shared with Specific Digital Output Function, e.g. PWM, SIO, UART.




***. Some specific functions switch I/O direction directly, not through PnM register.**

Bi-direction I/O Pin Shared with Specific Analog Input Function, e.g. ADC.




***. Some specific functions switch I/O direction directly, not through PnM register.**

# 5 CPU

SN8F5000 family is an enhanced 8051 microcontroller (MCU). It is fully compatible with MCS-51 instructions, hence the ability to cooperate with modern development environment (e.g. Keil C51). SN8F5000 CPU has 9.4 to 12.1 times faster than the original 8051 at the same frequency.

## 5.1 Memory Organization

SN8F5702 builds in two on-chip memories: internal RAM (IRAM) and program memory (IROM). The internal RAM is a 256-byte RAM which has higher access performance (direct and indirect addressing). The program memory is a 4 KB non-volatile memory and has a maximum 8 MHz speed limitation.


## 5.2 Internal RAM (IRAM)

256 X 8-bit RAM (Internal Data Memory)

<table>
<tr><th>Address</th><th colspan="2">RAM Location</th><th></th></tr>
<tr><td>000h<br>01Fh</td><td colspan="2">Work Register Area</td><td rowspan="3">00h-7Fh of RAM is direct and indirect access RAM</td></tr>
<tr><td>020h<br>02Fh</td><td colspan="2">Bit Addressable Area</td></tr>
<tr><td>030h<br>…<br>…<br>07Fh</td><td colspan="2">General Purpose Area</td></tr>
<tr><td>080h<br>…<br>…<br>…<br>0FFh</td><td>General Purpose Area (Indirect Access)</td><td>Special Function Register (Direct Access)</td><td>080h-0FFh store special function registers.<br><br>End of Bank 0</td></tr>
</table>

The 256-byte data RAM in internal data memory is a standard 8051 RAM access configuration. The upper 128-byte RAM is general purpose RAM and can configure by direct addressing access and indirect addressing access. The lower 128-byte can be indirect access RAM in general purpose or direct access RAM in special function register (SFR).

- 0x0000-0x007F: General purpose RAM contains work register area and bit addressable area. In this area, direct or indirect addressing can be used.
- 0x0000-0x001F: Work register area includes 4-bank. Each bank has 8 work registers (R0 - R7) which is selected by RS0/RS1 in PSW register.
- 0x0020-0x002F: Bit addressable area.

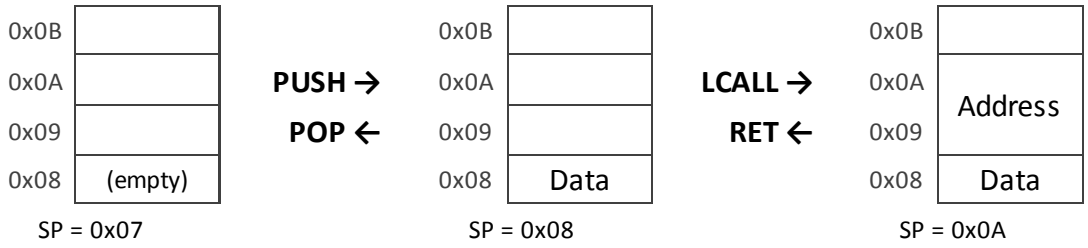
In the bit addressable area, user can read or write any single bit in this range by using the unique address for that bit. Supports 16bytes bit addressable RAM area giving 128 addressable bits. Each bit has individual address in the range from 00H to 7FH. Thus, the bit can be addressed directly. Bit0 of the byte 20H has bit address 00H and Bit 7 of the byte 20H has bit address 07H. Bit0 of the byte 2FH has bit address 78H and Bit 7 of the byte 2FH has bit address 7FH. When set “SETB 42H”, it means the bit2 of the byte 28H is set.

| | Byte Address | Bite 0 | Bite 1 | Bite 2 | Bite 3 | Bite 4 | Bite 5 | Bite 6 | Bite 7 |
|---|---|---|---|---|---|---|---|---|---|
| Bit Addressable Area | **0x20** | 0x00 | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 | 0x07 |
| | **0x21** | 0x08 | 0x09 | 0x0A | 0x0B | 0x0C | 0x0D | 0x0E | 0x0F |
| | **0x22** | 0x10 | 0x11 | 0x12 | 0x13 | 0x14 | 0x15 | 0x16 | 0x17 |
| | **0x23** | 0x18 | 0x19 | 0x1A | 0x1B | 0x1C | 0x1D | 0x1E | 0x1F |
| | **0x24** | 0x20 | 0x21 | 0x22 | 0x23 | 0x24 | 0x25 | 0x26 | 0x27 |
| | **0x25** | 0x28 | 0x29 | 0x2A | 0x2B | 0x2C | 0x2D | 0x2E | 0x2F |
| | **0x26** | 0x30 | 0x31 | 0x32 | 0x33 | 0x34 | 0x35 | 0x36 | 0x37 |
| | **0x27** | 0x38 | 0x39 | 0x3A | 0x3B | 0x3C | 0x3D | 0x3E | 0x3F |
| | **0x28** | 0x40 | 0x41 | 0x42 | 0x43 | 0x44 | 0x45 | 0x46 | 0x47 |
| | **0x29** | 0x48 | 0x49 | 0x4A | 0x4B | 0x4C | 0x4D | 0x4E | 0x4F |
| | **0x2A** | 0x50 | 0x51 | 0x52 | 0x53 | 0x54 | 0x55 | 0x56 | 0x57 |
| | **0x2B** | 0x58 | 0x59 | 0x5A | 0x5B | 0x5C | 0x5D | 0x5E | 0x5F |
| | **0x2C** | 0x60 | 0x61 | 0x62 | 0x63 | 0x64 | 0x65 | 0x66 | 0x67 |
| | **0x2D** | 0x68 | 0x69 | 0x6A | 0x6B | 0x6C | 0x6D | 0x6E | 0x6F |
| | **0x2E** | 0x70 | 0x71 | 0x72 | 0x73 | 0x74 | 0x75 | 0x76 | 0x77 |
| | **0x2F** | 0x78 | 0x79 | 0x7A | 0x7B | 0x7C | 0x7D | 0x7E | 0x7F |

- 0x0080~0x00FF: General purpose area in indirect addressing access or special function register in direct addressing access.

## 5.3 Stack

Stack can be assigned to any area of internal RAM (IRAM). However, it requires manual assignment to ensure its area does not overlap other RAM's variables. An overflow and underflow stack could also mistakenly overwrite other RAM's variables; thus, these factors should be considered while arrange the size of stack.




By default, stack pointer (SP register) points to 0x07 which means the stack area begin at IRAM address 0x08. In other word, if a planned stack area is assigned from IRAM address 0xC0, the appropriate SP register is anticipated to set at 0xBF after system reset.

An assembly PUSH instruction costs one byte of stack. LCALL, ACALL instructions and interrupt respectively costs two bytes stack. POP-instruction decreases one count, and a RET/RETI subtract two counts of stack pointer.

✷ ***Note: Stack and IRAM share the same area, Keil C51 compiler will not display "error" or "warning" when overlap condition is occurred so user must pay attention.***

## 5.4 Program Memory (IROM)

The program memory is a non-volatile storage area where stores code, look-up ROM table, and other data with occasional modification. It can be updated by debug tools like SN-Link3, and a program can also self-update via in-system program process (refer to In-system Program).

| Address | ROM | Comment |
|---|---|---|
| 0000H | Reset vector | Reset vector |
| 0001H<br>0002H | General purpose area | User program |
| 0003H | **INT0 Interrupt vector** | **Interrupt vector** |
| 000BH | **TIMER0 Interrupt vector** | |
| 001BH | **TIMER1 Interrupt vector** | |
| 0023H | **UART Interrupt vector** | |
| 002BH | **TIMER2 Interrupt vector** | |
| 0043H | **I2C Interrupt vector** | |
| 004BH | **SPI Interrupt vector** | |
| 0053H | **T2COM0 Interrupt vector** | |
| 005BH | **T2COM1 Interrupt vector** | |
| 0063H | **T2COM2 Interrupt vector** | |
| 006BH | **T2COM3 Interrupt vector** | |
| 0083H | **PWM1 Interrupt vector** | |
| 008BH | **ADC Interrupt vector** | |
| 008CH<br>.<br>.<br>.<br>. | General purpose area | User program<br><br><br><br>End of user program |
| 0FF6H<br>0FF7H<br>.<br>0FFEH<br>0FFFH | Reserved | |

The ROM includes reset vector, Interrupt vector, general purpose area and reserved area. The reset vector is program beginning address. The interrupt vector is the head of interrupt service routine when any interrupt occurring. The general purpose area is main program area including main loop, sub-routines and data table.

- 0x0000 Reset vector: Program counter points to 0x0000 after any reset events (power on reset, reset pin reset, watchdog reset, LVD reset…).
- 0x0001~0x0002: General purpose area to process system reset operation.
- 0x0003~0x008B: Multi interrupt vector area. Each of interrupt events has a unique interrupt vector.
- 0x008C~0x0FDF: General purpose area for user program and ISP (EEPROM function).

- 0x0FE0~0x0FF5: General purpose area for user program. Do not execute ISP.
- 0x0FF6~0x0FFF: Reserved area. Do not execute ISP.

## 5.5 Program Memory Security

The SN8F5702 provides security options at the disposal of the designer to prevent unauthorized access to information stored in FLASH memory. When enable security option, the ROM code is secured and not dumped complete ROM contents. ROM security rule is all address ROM data protected and outputs 0x00.

## 5.6 Data Pointer

A data pointer helps to specify the IROM address while performing MOVC instructions. The microcontroller has one set of data pointer (DPH/DPL).

## 5.7 Stack and Data Pointer Register

| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| SP | SP7 | SP6 | SP5 | SP4 | SP3 | SP2 | SP1 | SP0 |
| DPL | DPL7 | DPL6 | DPL5 | DPL4 | DPL3 | DPL2 | DPL1 | DPL0 |
| DPH | DPH7 | DPH6 | DPH5 | DPH4 | DPH3 | DPH2 | DPH1 | DPH0 |

**SP Register (0x81)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | SP | R/W | 0x07 | Stack pointer |

**DPL Register (0x82)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | DPL[7:0] | R/W | 0x00 | Low byte of DPTR0 |

**DPH Register (0x83)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | DPH[7:0] | R/W | 0x00 | High byte of DPTR0 |

# 6 Special Function Registers

## 6.1 Special Function Register Memory Map

| BIN / HEX | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| F8 | - | P0M | P1M | P2M | - | - | - | PFLAG |
| F0 | B | P0UR | P1UR | P2UR | - | - | - | SRST |
| E8 | - | - | - | - | - | - | - | - |
| E0 | ACC | SPSTA | SPCON | SPDAT | P0OC | CLKSEL | CLKCMD | TCON0 |
| D8 | S0CON2 | - | I2CDAT | I2CADR | I2CCON | I2CSTA | SMBSEL | SMBDST |
| D0 | PSW | IEN4 | ADM | ADB | ADR | VREFH | P1CON | - |
| C8 | T2CON | - | CRCL | CRCH | TL2 | TH2 | - | - |
| C0 | IRCON | CCEN | CCL1 | CCH1 | CCL2 | CCH2 | CCL3 | CCH3 |
| B8 | IEN1 | IP1 | S0RELH | PW1DL | PW1DH | PW1A | PW1CH | IRCON2 |
| B0 | - | - | - | - | - | - | - | - |
| A8 | IEN0 | IP0 | S0RELL | PW1M | PW1YL | PW1YH | PW1BL | PW1BH |
| A0 | P2 | - | - | - | - | - | - | - |
| 98 | S0CON | S0BUF | IEN2 | - | - | P0CON | P2CON | - |
| 90 | P1 | P1W | - | - | PECMD | PEROML | PEROMH | PERAM |
| 88 | TCON | TMOD | TL0 | TL1 | TH0 | TH1 | CKCON | PEDGE |
| 80 | P0 | SP | DPL | DPH | - | - | WDTR | PCON |

✷ ***Note: All SFRs in the left-most column are bit-addressable. (Every 0x0/0x8-ending SFR addresses are bit-addressable).***

## 6.2 Special Function register Description

**0x80 - 0x9F Registers Description**

| Register | Address | Description |
|---|---|---|
| P0 | 0x80 | Port 0 data buffer. |
| SP | 0x81 | Stack pointer register. |
| DPL | 0x82 | Data pointer 0 low byte register. |
| DPH | 0x83 | Data pointer 0 high byte register. |
| - | 0x84 | - |
| - | 0x85 | - |
| WDTR | 0x86 | Watchdog timer clear register. |
| PCON | 0x87 | System mode register. |
| TCON | 0x88 | Timer 0 / 1 controls register. |
| TMOD | 0x89 | Timer 0 / 1 mode register. |
| TL0 | 0x8A | Timer 0 counting low byte register. |
| TL1 | 0x8B | Timer 1 counting low byte register. |
| TH0 | 0x8C | Timer 0 counting high byte register. |
| TH1 | 0x8D | Timer 1 counting high byte register. |
| CKCON | 0x8E | Extended cycle controls register. |
| PEDGE | 0x8F | External interrupt edge controls register. |
| P1 | 0x90 | Port 1 data buffer. |
| P1W | 0x91 | Port 1 wake-up controls register. |
| - | 0x92 | - |
| - | 0x93 | - |
| PECMD | 0x94 | In-System Program command register. |
| PEROML | 0x95 | In-System Program ROM address low byte |
| PEROMH | 0x96 | In-System Program ROM address high byte |
| PERAM | 0x97 | In-System Program RAM mapping address |
| S0CON | 0x98 | UART control register. |
| S0BUF | 0x99 | UART data buffer. |
| IEN2 | 0x9A | Interrupts enable register |
| - | 0x9B | - |
| - | 0x9C | - |
| P0CON | 0x9D | Port 0 configuration controls register. |
| P2CON | 0x9E | Port 2 configuration controls register. |
| - | 0x9F | - |

**0xA0 - 0xBF Registers Description**

| Register | Address | Description |
|---|---|---|
| P2 | 0xA0 | Port 2 data buffer |
| - | 0xA1 | - |
| - | 0xA2 | - |
| - | 0xA3 | - |
| - | 0xA4 | - |
| - | 0xA5 | - |
| - | 0xA6 | - |
| - | 0xA7 | - |
| IEN0 | 0xA8 | Interrupts enable register |
| IP0 | 0xA9 | Interrupts priority register. |
| S0RELL | 0xAA | UART reload low byte register. |
| PW1M | 0xAB | PW1 controls register. |
| PW1YL | 0xAC | PW1 cycle controls buffer low byte. |
| PW1YH | 0xAD | PW1 cycle controls buffer high byte. |
| PW1BL | 0xAE | PW1 B point dead band controls buffer low byte. |
| PW1BH | 0xAF | PW1 B point dead band controls buffer high byte. |
| - | 0xB0 | - |
| - | 0xB1 | - |
| - | 0xB2 | - |
| - | 0xB3 | - |
| - | 0xB4 | - |
| - | 0xB5 | - |
| - | 0xB6 | - |
| - | 0xB7 | - |
| IEN1 | 0xB8 | Interrupts enable register |
| IP1 | 0xB9 | Interrupts priority register. |
| S0RELH | 0xBA | UART reload high byte register. |
| PW1DL | 0xBB | PW1 duty controls buffer low byte. |
| PW1DH | 0xBC | PW1 duty controls buffer high byte. |
| PW1A | 0xBD | PW1 A point dead band controls buffer. |
| PW1CH | 0xBE | PW1 channel control buffer. |
| IRCON2 | 0xBF | Interrupts request register. |

**0xC0 - 0xDF Registers Description**

| Register | Address | Description |
|---|---|---|
| IRCON | 0xC0 | Interrupts request register. |
| CCEN | 0xC1 | Timer 2 Compare /capture enable register. |
| CCL1 | 0xC2 | Timer 2 Compare /capture module 1 low byte register. |
| CCH1 | 0xC3 | Timer 2 Compare /capture module 1 high byte register. |
| CCL2 | 0xC4 | Timer 2 Compare /capture module 2 low byte register. |
| CCH2 | 0xC5 | Timer 2 Compare /capture module 2 high byte register. |
| CCL3 | 0xC6 | Timer 2 Compare /capture module 3 low byte register. |
| CCH3 | 0xC7 | Timer 2 Compare /capture module 3 high byte register. |
| T2CON | 0xC8 | Timer 2 controls register. |
| - | 0xC9 | - |
| CRCL | 0xCA | Timer 2 Compare/capture module 0 & reload function low byte register. |
| CRCH | 0xCB | Timer 2 Compare/capture module 0 & reload function high byte register. |
| TL2 | 0xCC | Timer 2 counting low byte register. |
| TH2 | 0xCD | Timer 2 counting high byte register. |
| - | 0xCE | - |
| - | 0xCF | - |
| PSW | 0xD0 | System flag register. |
| IEN4 | 0xD1 | Interrupts enable register |
| ADM | 0xD2 | ADC controls register. |
| ADB | 0xD3 | ADC data buffer. |
| ADR | 0xD4 | ADC resolution selects register. |
| VREFH | 0xD5 | ADC reference voltage controls register. |
| P1CON | 0xD6 | Port 1 configuration controls register. |
| - | 0xD7 | - |
| S0CON2 | 0xD8 | UART baud rate controls register. |
| - | 0xD9 | - |
| I2CDAT | 0xDA | I2C data buffer. |
| I2CADR | 0xDB | Own I2C slave address. |
| I2CCON | 0xDC | I2C interface operation control register. |
| I2CSTA | 0xDD | I2C Status Code. |
| SMBSEL | 0xDE | SMBus mode controls register. |
| SMBDST | 0xDF | SMBus internal timeout register. |

**0xE0 - 0xFF Registers Description**

| Register | Address | Description |
|---|---|---|
| ACC | 0xE0 | Accumulator register. |
| SPSTA | 0xE1 | SPI statuses register. |
| SPCON | 0xE2 | SPI control register. |
| SPDAT | 0xE3 | SPI data buffer. |
| P0OC | 0xE4 | Open drain controls register. |
| CLKSEL | 0xE5 | Clock switch selects register. |
| CLKCMD | 0xE6 | Clock switch controls Register. |
| TCON0 | 0xE7 | Timer 0 / 1 clock controls register. |
| - | 0xE8 | - |
| - | 0xE9 | - |
| - | 0xEA | - |
| - | 0xEB | - |
| - | 0xEC | - |
| - | 0xED | - |
| - | 0xEE | - |
| - | 0xEF | - |
| B | 0xF0 | Multiplication/ division instruction data buffer. |
| P0UR | 0xF1 | Port 0 pull-up resister controls register. |
| P1UR | 0xF2 | Port 1 pull-up resister controls register. |
| P2UR | 0xF3 | Port 2 pull-up resister controls register. |
| - | 0xF4 | - |
| - | 0xF5 | - |
| - | 0xF6 | - |
| SRST | 0xF7 | Software reset controls register. |
| - | 0xF8 | - |
| P0M | 0xF9 | Port 0 input/output mode register. |
| P1M | 0xFA | Port 1 input/output mode register. |
| P2M | 0xFB | Port 2 input/output mode register. |
| - | 0xFC | - |
| - | 0xFD | - |
| - | 0xFE | - |
| PFLAG | 0xFF | Reset flag register. |

## 6.3 System Registers

| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| ACC | ACC7 | ACC6 | ACC5 | ACC4 | ACC3 | ACC2 | ACC1 | ACC0 |
| B | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PSW | CY | AC | F0 | RS1 | RS0 | OV | F1 | P |

**ACC Register (0xE0)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | ACC[7:0] | R/W | 0x00 | The ACC is an 8-bit data register responsible for transferring or manipulating data between ALU and data memory. If the result of operating is overflow (OV) or there is carry (C or AC) and parity (P) occurrence, then these flags will be set to PSW register. |

**B Register (0xF0)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | B[7:0] | R/W | 0x00 | The B register is used during multiplying and division instructions. It can also be used as a scratch-pad register to hold temporary data. |

**PSW Register (0xD0)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | CY | R/W | 0 | Carry flag.<br>0: Addition without carry, subtraction with borrowing signal, rotation with shifting out logic "0", comparison result < 0.<br>1: Addition with carry, subtraction without borrowing, rotation with shifting out logic "1", comparison result ≥ 0. |
| 6 | AC | R/W | 0 | Auxiliary carry flag.<br>0: If there is no a carry-out from 3rd bit of Accumulator in BCD operations.<br>1: If there is a carry-out from 3rd bit of Accumulator in BCD operations. |
| 5 | F0 | R/W | 0 | General purpose flag 0. General purpose flag available for user. |
| 4..3 | RS[1:0] | R/W | 00 | Register bank select control bit, used to select working register bank.<br>00: 00H – 07H (Bnak0)<br>01: 08H – 0FH (Bnak1)<br>10: 10H – 17H (Bnak2)<br>11: 18H – 1FH (Bnak3) |
| 2 | OV | R/W | 0 | Overflow flag.<br>0: Non-overflow in Accumulator during arithmetic Operations.<br>1: overflow in Accumulator during arithmetic Operations. |
| 1 | F1 | R/W | 0 | General purpose flag 1. General purpose flag available for user. |
| 0 | P | R | 0 | Parity flag. Reflects the number of '1's in the Accumulator.<br>0: if Accumulator contains an even number of '1's.<br>1: Accumulator contains an odd number of '1's. |

## 6.4 Register Declaration

SN8F5702 has many registers to control various functions, but SFR name is not predefined in the C51 / A51 compiler. To make programming easier and therefore need to add header files to declare SFR name.

When using the assembly code programs, please add the following sentence.

```
1 $NOMOD51 ;Do not recognize the 8051-specific predefined special register.
2 #include <SN8F5702.H>
```

When using the C code programs, please add the following sentence.

```
1 #include <SN8F5702.H>
```

After adding the header file, user can use name of registers to program. During compilation, the compiler will register name translate into register position through the header file.

Different devices need to use a different header file to declare, but the option file is to use the same.

| Device | Header file | Options file |
|---|---|---|
| SN8F5702 | SN8F5702.h | OPTIONS_SN8F5702.A51 |
| SN8F570200 | SN8F570200.h | |
| SN8F570202 | SN8F570202.h | |
| SN8F570210 | SN8F570210.h | |
| SN8F570211 | SN8F570211.h | |
| SN8F570212 | SN8F570212.h | |
| SN8F570213 | SN8F570213.h | |

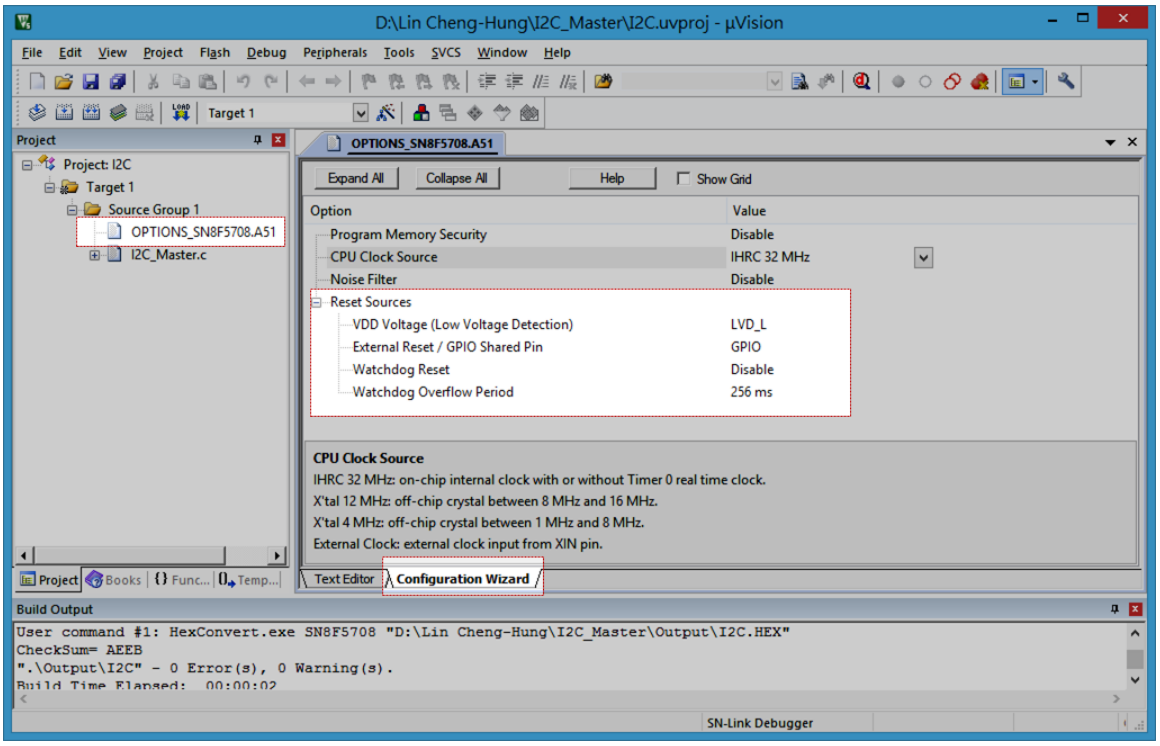
# 7 Reset and Power-on Controller

The reset and power-on controller has four reset sources: low voltage detectors (LVDs), watchdog, programmable external reset pin, and software reset. The first three sources would trigger an additional power-on sequence. Subsequently, the microcontroller initializes all registers and starts program execution with its reset vector (ROM address 0x0000).

## 7.1 Configuration of Reset and Power-on Controller

SONiX publishes an *OPTIONS_SN8F5702.A51 file* in *SN-Link Driver for Keil C51.exe* (downloadable on cooperative website: www.sonix.com.tw). This *options file* contains appropriate parameters of reset sources and CPU clock source selection, and is strongly recommended to add to Keil project. *SN8F5000 Debug Tool Manual* provides the further detail of this configuration.

- Program Memory Security
- CPU Clock Source
- Reset Source : VDD Voltage (Low Voltage Detection)
- Reset Source : External Reset / GPIO Shared Pin
- Reset Source : Watchdog Reset & Overflow Period

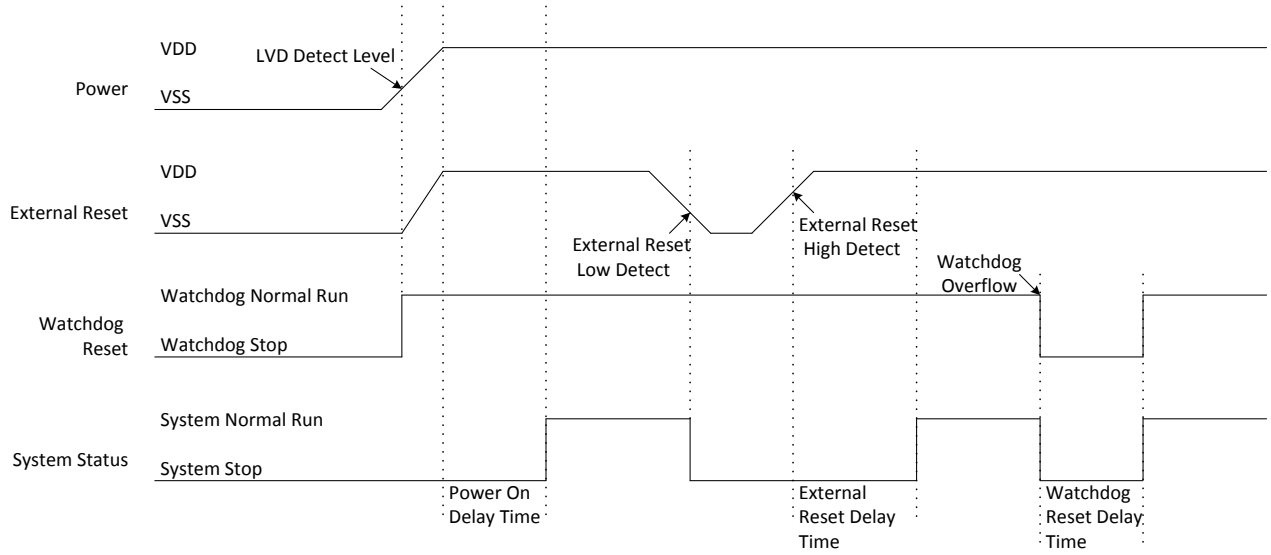


The code option is the system hardware configurations including oscillator type, noise filter option, watchdog timer operation, LVD option, reset pin option and flash ROM security control. The code option items are as following table:

| Code Option | Content | Function Description |
|---|---|---|
| Program Memory Security | Security Disable | Disable ROM code Security function |
| | Security Enable | Enable ROM code Security function |
| CPU Clock Source | IHRC 32MHz | High speed internal 32MHz RC. |
| LVD | LVD_L | LVD will reset chip if VDD is below 1.8V |
| External Reset | Reset with De-bounce | Enable External reset pin with De-bounce |
| | Reset without De-bounce | Enable External reset pin without De-bounce |
| | GPIO with P02 | Enable P02 |
| Watchdog Reset | Always | Watchdog timer is always on enable even in STOP mode and IDLE mode |
| | Enable | Enable watchdog timer. Watchdog timer stops in STOP mode and IDLE mode |
| | Disable | Disable Watchdog function |
| Watchdog Overflow Period | 64ms | Watchdog timer clock source $F_{ILRC}$ /4 |
| | 128ms | Watchdog timer clock source $F_{ILRC}$ /8 |
| | 256ms | Watchdog timer clock source $F_{ILRC}$ /16 |
| | 512ms | Watchdog timer clock source $F_{ILRC}$ /32 |

## 7.2 Power-on Sequence

A power-on sequence would be triggered by LVD, watchdog, and external reset pin. It takes place between the end of reset signal and program execution. Overall, it includes two stages: power stabilization period, and clock stabilization period.

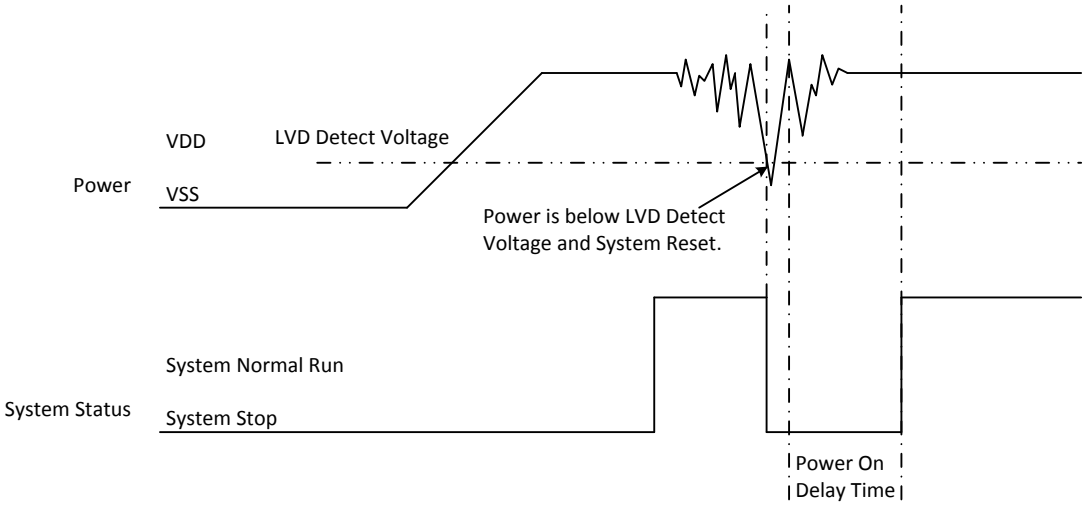



The power stabilization period spends 4.6ms in typical condition. Afterward the microcontroller fetches CPU Clock Source selection automatically. The selected clock source would be driven, and the system counts 2048 times of the clock period and 5 times of the internal low-speed oscillator clocks to ensure its reliability.

✷ ***Note: In high power noise environment, user can put 10ohm resistor in the front of 0.1uF capacitor & VDD PAD to suppress power noise and avoid IC damage.***

## 7.3 LVD Reset

The low voltage detectors monitor VDD pin's voltage at only one level: 1.8 V. Depend on low voltage detection configuration, the comparison result can be seen as a system reset signal. The table below lists low voltage detection configuration, LVD_L, and the results of VDD pin's condition.




| Condition | LVD_L |
|---|---|
| VDD ≤ 1.8 V | Reset |

## 7.4 Watchdog Reset

Watchdog is a periodic reset signal generator for the purpose of monitoring the execution flow. Its internal timer is expected to be cleared in a check point of program flow; therefore, the actual reset signal would be generated only after a software problem occurs. Writing 0x5A to WDTR is the proper method to place a check point in program.

```
1     WDTR = 0x5A;
```

Watchdog timer interval time = 256 * 1/ (Internal Low-Speed oscillator frequency/WDT Pre-scalar)
= 256 / ($F_{ILRC}$/WDT Pre-scaler) …sec

| Internal low-speed oscillator | WDT pre-scaler | Watchdog interval time |
|---|---|---|
| $F_{ILRC}$=16 kHz | $F_{ILRC}$ /4 | 256/(16000/4)=64ms |
| | $F_{ILRC}$ /8 | 256/(16000/8)=128ms |
| | $F_{ILRC}$ /16 | 256/(16000/16)=256ms |
| | $F_{ILRC}$ /32 | 256/(16000/32)=512ms |

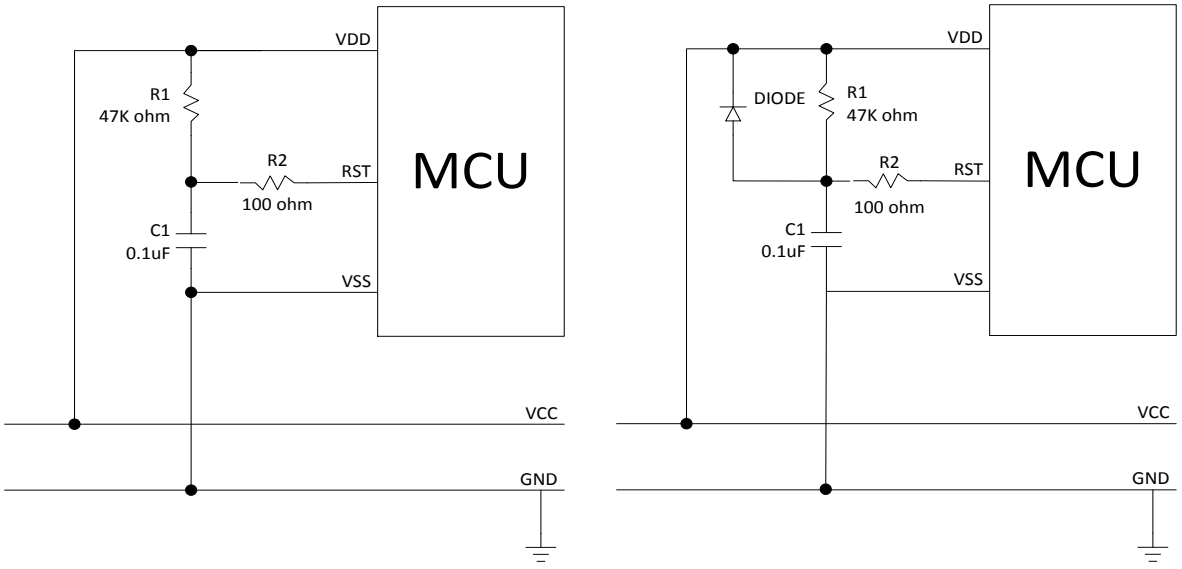
The operation mode of watchdog is configurable in options file:
**Always mode** counts its internal timer in all CPU operation modes (normal, IDLE, SLEEP);
**Enable mode** counts its internal timer during CPU stays in normal mode, and it would not trigger watchdog reset in IDLE and STOP modes;
**Disable mode** suspends its internal timer at all CPU modes, and the watchdog would not trigger in this condition.

When watchdog is operating in always mode, the system will consume additional power.

## 7.5 External Reset Pin

Programmable external reset pin is configurable in *options file*. Once it is enabled, it monitors its shared pin's logic level. A logical low (lower than 30% of VDD) would immediately trigger system reset until the input is recovered to high (lager than 70% of VDD).

An optional de-bounce period can improve reset signal's stability. Instead of immediate reset, the system reset requires an 8-ms-long logic low to avoid bouncing from a button key. Any signal lower than de-bounce period would not affect the CPU's execution.




✷ ***Note:***

***1. The reset circuit is no any protection against unusual power or brown out reset on the left side of the figure.***

***2. The R2 100 ohm resistor of "Simply reset circuit" and "Diode & RC reset circuit" is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS) on the right side of the figure.***

## 7.6 Software Reset

A software reset would be generated after consecutively set SRSTREQ register. As a result, this procedure enables firmware's ability to reset microcontroller (e.g. reset after firmware update). The following sample C code repeatedly set the least bit of SRST register to perform software reset.

```
1    SRST = 0x01;
2    SRST = 0x01;
```

## 7.7 Reset and Power-on Controller Registers

| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PFLAG | POR | WDT | RST | - | - | - | - | - |
| SRST | - | - | - | - | - | - | - | SRSTREQ |
| WDTR | WDTR7 | WDTR6 | WDTR5 | WDTR4 | WDTR3 | WDTR2 | WDTR1 | WDTR0 |

**PFLAG Register**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | POR | R | - | This bit is automatically set if the microcontroller has been reset by LVD. |
| 6 | WDT | R | - | This bit is automatically set if the microcontroller has been reset by watchdog. |
| 5 | RST | R | - | This bit is automatically set if the microcontroller has been reset by external reset pin. |
| 4..0 | Reserved | R | 0 | |

**SRST Register**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..1 | Reserved | R | 0 | |
| 0 | SRSTREQ | R/W | 0 | Read: This bit is automatically set if the microcontroller has been reset by software reset.<br>Write: Consecutively set this bit for two times to trigger software reset. |

**WDTR Register (0x86)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | WDTR[7:0] | W | - | Watchdog clear is controlled by WDTR register. Moving 0x5A data into WDTR is to reset watchdog timer. |

# 8 System Clock and Power Management

For power saving purpose, the microcontroller built in three different operation modes: normal, IDLE, and STOP mode.

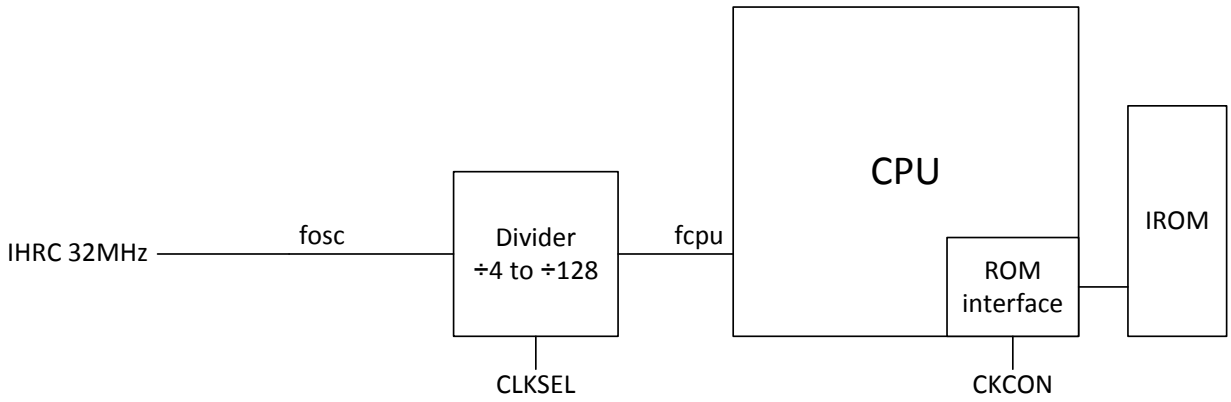
The normal mode means that CPU and peripheral functions are under normally execution. The system clock is based on the combination of source selection, clock divider, and program memory wait state. IDLE mode is the situation that temporarily suspends CPU clock and its execution, yet it remains peripherals' functionality (e.g. timers, PWM, SPI, UART, and I2C). STOP mode disables all functions and clock generator until a wakeup signal to return normal mode.

## 8.1 System Clock

The microcontroller includes an on-chip clock generator (IHRC 32MHz). The reset and power-on controller automatically loads clock source selection during power-on sequence. Therefore, the selected clock source is seen as 'fosc' domain which is a fixed frequency at any time.

Subsequently, the selected clock source (fosc) is divided by 4 to 128 times which is controlled by CLKSEL register. The CPU input the divided clock as its operation base (named fcpu). Applying CLKSEL's setting when CLKCMD register be written 0x69.

```
1     CKCON = 0x70;     // For change safely the system clock
2     CLKSEL = 0x05;    // Set fcpu = fosc / 4
3     CLKCMD = 0x69;    // Apply CLKSEL's setting
4     CKCON = 0x00;     // IROM fetch = fcpu / 1
```




ROM interface is built in between CPU and IROM (program memory). It optionally extends the data fetching cycle in order to support lower speed program memory.

**IROM fetching cycle (Instruction cycle) ≦ 8MHz**

✷ ***Note: For user develop program in C language or assembly, the first line of the program "must be set" CLKSEL= 0x05~0x00, CLKMD= 0x69 and then set CKCON= 0x00~0x70, this priority cannot be modified.***

System clock rate and program memory extended cycle limitation as follows.

| Code Option<br>CPU Clock Source | Fcpu = CLKSEL[2:0] | IROM Fetch = CKCON[6:4] |
|---|---|---|
| IHRC 32M | **Only Support**<br>000 = fosc / 128<br>001 = fosc / 64<br>010 = fosc / 32<br>011 = fosc / 16<br>100 = fosc / 8<br>101 = fosc / 4 | **Support**<br>**000 = fcpu / 1 => Recommend!**<br>001 = fcpu / 2<br>010 = fcpu / 3<br>011 = fcpu / 4<br>100 = fcpu / 5<br>101 = fcpu / 6<br>110 = fcpu / 7<br>111 = fcpu / 8 |

## 8.2 High Speed Clock and Real time clock

High-speed clock only has internal clock. The internal high-speed oscillator is 32MHz RC type.

- IHRC 32M: The system high-speed clock source is internal high-speed 32MHz RC type oscillator.

## 8.3 Power Management

After the end of reset signal and power-on sequence, the CPU starts program execution at the speed of fcpu. Overall, the CPU and all peripherals are functional in this situation (categorized as normal mode).

The least two bits of PCON register (IDLE at bit 0 and STOP at bit 1) control the microcontroller's power management unit.

If IDLE bit is set by program, only CPU clock source would be gated. Consequently, peripheral functions (such as timers, PWM, and I2C) and clock generator (IHRC 32 MHz) remain execution in this status. Any change from P0/P1 input and interrupt events can make the microcontroller turns back to normal mode, and the IDLE bit would be cleared automatically.

- Any function can work in IDLE mode. Only CPU is suspended.
- The IDLE mode wake-up sources are P0/P1 level change trigger and any interrupt event.

If STOP bit is set, by contrast, CPU, peripheral functions, and clock generator are suspended. Data storage in registers and RAM would be kept in this mode. Any change from P0/P1 can wake up the microcontroller and resume system's execution. STOP bit would be cleared automatically.

- CPU, peripheral functions, and clock generator are suspended.
- The STOP mode wake-up source is P0/P1 level change trigger.

For user who is develop program in C language, IDLE and STOP macros is strongly recommended to control the microcontroller's system mode, instead of set IDLE and STOP bits directly.

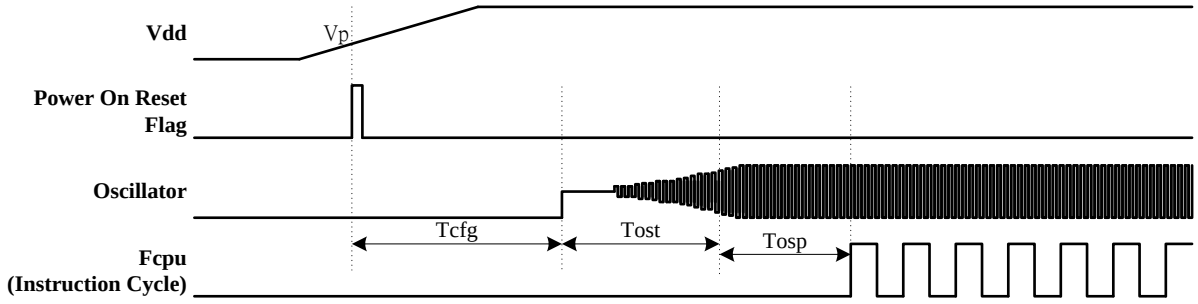
```
1     IDLE();
2     STOP();
```

✷ ***Note: Into IDLE mode or STOP mode by "Assembly Language" must be using MOV instruction.***

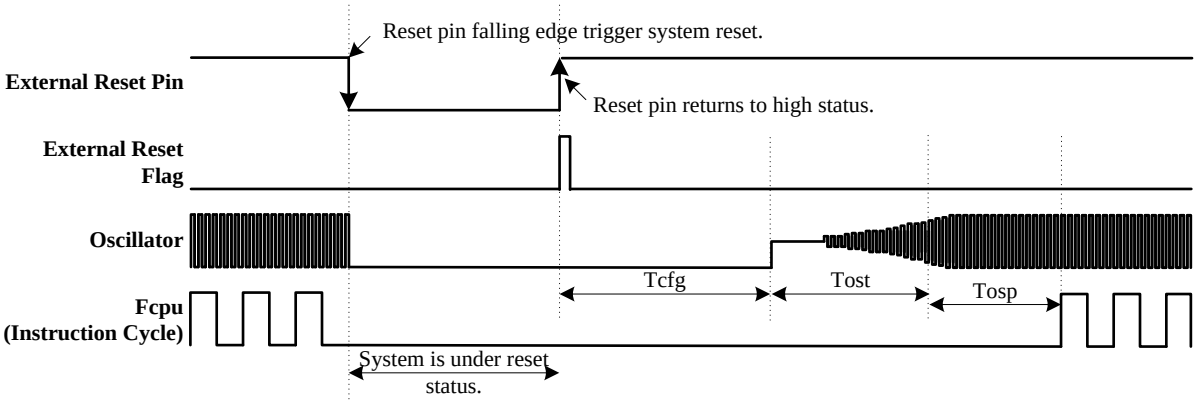
## 8.4 System Clock Timing

| Parameter | Symbol | Description | Typical |
|---|---|---|---|
| Hardware configuration time | Tcfg | $8*F_{ILRC} + 2^{17}*F_{IHRC}$ | 4.6ms @ $F_{ILRC}$ = 16KHz & $F_{IHRC}$ = 32MHz |
| Oscillator start up time | Tost | The start-up time is depended on oscillator's material, factory and architecture. Normally, the low-speed oscillator's start-up time is lower than high-speed oscillator. The RC type oscillator's start-up time is faster than crystal type oscillator. | - |
| Oscillator warm-up time | Tosp | Oscillator warm-up time of reset condition.<br>$2048*F_{hosc}+5*F_{ILRC}$<br>(Power on reset, LVD reset, watchdog reset, external reset pin active.) | 377us @ $F_{hosc}$ = 32MHz |
| | | Oscillator warm-up time of power down mode wake-up condition.<br>$64*F_{hosc}+5*F_{ILRC}$ ……RC type oscillator, e.g. internal high-speed RC type oscillator. | RC:<br>315us @ $F_{hosc}$ = 32MHz |

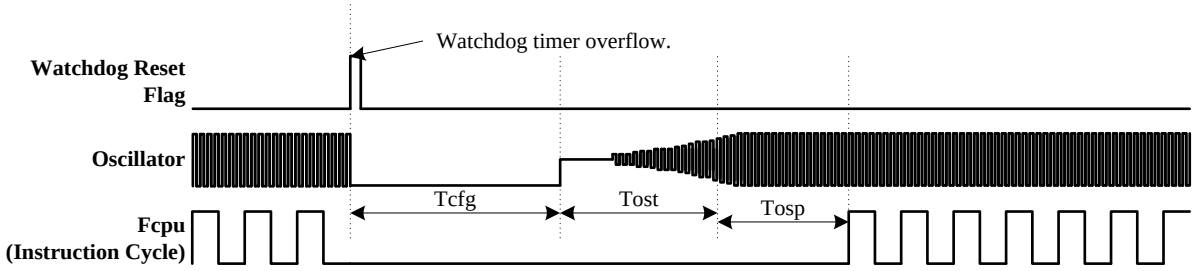
- Power On Reset Timing



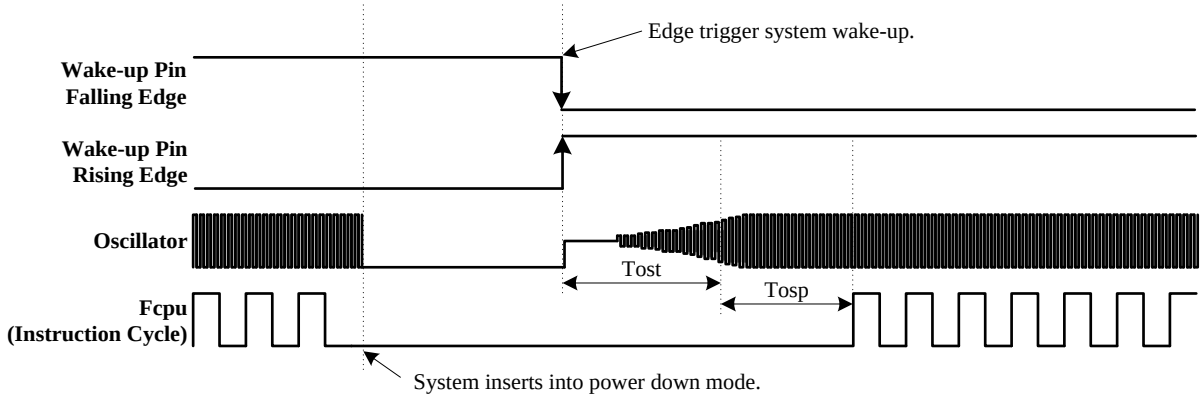

- External Reset Pin Reset Timing



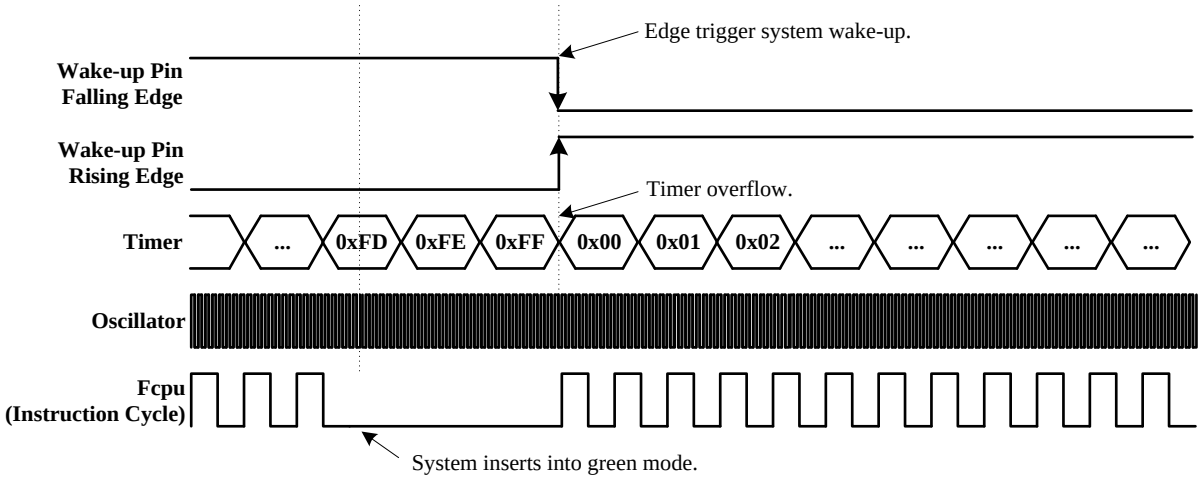

- Watchdog Reset Timing




- STOP Mode Wake-up Timing

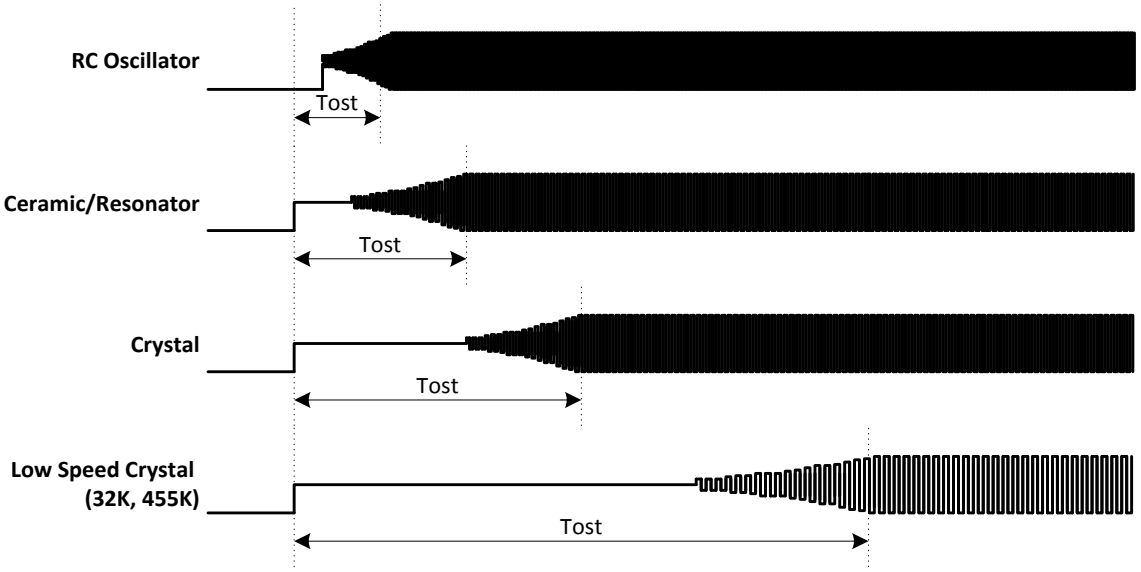


- IDLE Mode Wake-up Timing




- Oscillator Start-up Time

The start-up time is depended on oscillator's material, factory and architecture. Normally, the low-speed oscillator's start-up time is lower than high-speed oscillator.



## 8.5 System Clock and Power Management Registers

| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| CKCON | - | PWSC2 | PWSC1 | PWSC0 | ESYN | EWSC2 | EWSC1 | EWSC0 |
| CLKSEL | - | - | - | - | - | CLKSEL2 | CLKSEL1 | CLKSEL0 |
| CLKCMD | CMD7 | CMD6 | CMD5 | CMD4 | CMD3 | CMD2 | CMD1 | CMD0 |
| PCON | SMOD | - | - | - | - | GF0 | STOP | IDLE |
| P1W | P17W | P16W | P15W | P14W | P13W | P12W | P11W | P10W |

**CKCON Register (0x8E)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | Reserved | R | 0 | |
| 6..4 | PWSC[2:0] | R/W | 111 | Extended cycle(s) applied to reading program memory<br>000: non<br>001: 1 cycle<br>010: 2 cycles<br>011: 3 cycles<br>100: 4 cycles<br>101: 5 cycles<br>110: 6 cycles<br>111: 7 cycles |
| Else | Reserved | R | 0001 | |

**CLKCMD Register (0xE6)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | CMD[7:0] | W | 0x00 | Writing 0x69 to apply CLKSEL's setting. |

**CLKSEL Register (0xE5)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..3 | Reserved | R | 0x00 | |
| 2..0 | CLKSEL[2:0] | R/W | 111 | CLKSEL would be applied by writing CLKCMD.<br>000: fcpu = fosc / 128<br>001: fcpu = fosc / 64<br>010: fcpu = fosc / 32<br>011: fcpu = fosc / 16<br>100: fcpu = fosc / 8 |

| | | | | |
|---|---|---|---|---|
| | | | | 101: fcpu = fosc / 4<br>Others: Reserved. |

**PCON Register (0x87)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | | | | Refer to other chapter(s) |
| 6..3 | Reserved | R | 0x00 | |
| 2 | GF0 | R/W | 0 | General Purpose Flag |
| 1 | STOP | R/W | 0 | 1: Microcontroller switch to STOP mode |
| 0 | IDLE | R/W | 0 | 1: Microcontroller switch to IDLE mode |

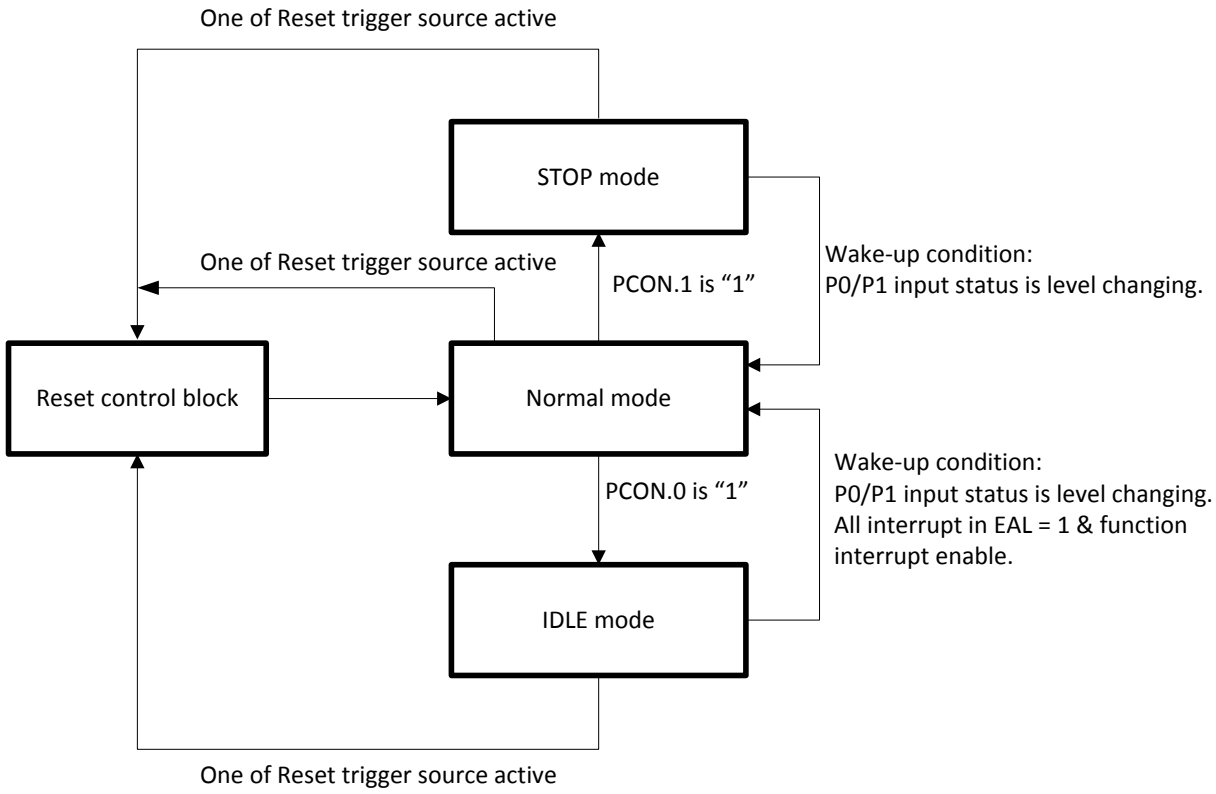
**P1W Register (0x91)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | P1nW | R/W | 0 | 0: Disable P1.n wakeup functionality<br>1: Enable P1.n wakeup functionality |

# 9 System Operating Mode

The chip builds in three operating mode for difference clock rate and power saving reason. These modes control oscillators, op-code operation and analog peripheral devices' operation.

- Normal mode: System high-speed operating mode
- IDLE mode: System idle mode (Green mode)
- STOP mode: System power saving mode (Sleep mode)



The operating mode clock control as following table:

| Operating Mode | Normal Mode | IDLE Mode | STOP Mode |
|---|---|---|---|
| IHRC | IHRC: Running | IHRC: Running | Stop |
| ILRC | Running | Running | Watchdog always: Running<br>Other : stop |
| CPU instruction | Executing | Stop | Stop |
| Timer 0<br>(Timer, Event counter) | Active by TR0 | Active by TR0 | Inactive |
| Timer 1<br>(Timer, Event counter) | Active by TR1 | Active by TR1 | Inactive |
| Timer 2<br>(Timer, capture, T2COM) | Active as enable | Active as enable | Inactive |
| PWM | Active as enable | Active as enable | Inactive |
| UART | Active as enable | Active as enable | Inactive |
| SPI | Active as enable | Active as enable | Inactive |
| I2C | Active as enable | Active as enable | Inactive |
| ADC | Active as enable | Active as enable | Inactive |
| Watchdog timer | By Watchdog<br>Code option | By Watchdog<br>Code option | By Watchdog<br>Code option |
| Internal interrupt | All active | All active | All inactive |
| External interrupt | All active | All active | All inactive |
| Wakeup source | - | P0, P1, Reset, All interrupt in EAL = 1 & function interrupt enable | P0, P1, Reset |

- IHRC: Internal high-speed oscillator RC type.
- ILRC: Internal low-speed oscillator RC type.

## 9.1 Normal Mode

The Normal Mode is system high clock operating mode. The system clock source is from high speed oscillator. The program is executed. After power on and any reset trigger released, the system inserts into normal mode to execute program. When the system is wake-up from STOP/IDLE mode, the system also inserts into normal mode. In normal mode, the high speed oscillator is active, and the power consumption is largest of all operating modes.

- The program is executed, and full functions are controllable.
- The system rate is high speed.
- The high speed oscillator and internal low speed RC type oscillator are active.
- Normal mode can be switched to other operating modes through PCON register.
- STOP/IDLE mode is wake-up to normal mode.

## 9.2 STOP Mode

The STOP mode is the system ideal status. No program execution and oscillator operation. Only internal regulator is active to keep all control gates status, register status and SRAM contents. The STOP mode is waked up by P0/P1 hardware level change trigger. P0 wake-up function is always enables. The STOP mode is wake-up to normal mode. Inserting STOP mode is controlled by stop bit of PCON register. When stop = 1, the system inserts into STOP Mode. After system wake-up from STOP mode, the stop bit is disabled (zero status) automatically.

- The program stops executing, and full functions are disabled.
- All oscillators including external high speed oscillator, internal high speed oscillator and internal low speed oscillator stop.
- Only internal regulator is active to keep all control gates status, register status and SRAM contents.
- The system inserts into normal mode after wake-up from STOP mode.
- The STOP mode wake-up source is P0/P1 level change trigger.

## 9.3 IDLE Mode

The IDLE mode is another system ideal status not like STOP mode. In STOP mode, all functions and hardware devices are disabled. But in IDLE mode, the system clock source keeps running, so the power consumption of IDLE mode is larger than STOP mode. In IDLE mode, the program isn't executed, but the timer with wake-up function is active as enabled, and the timer clock source is the non-stop system clock. The IDLE mode has 2 wake-up sources. One is the P0/P1 level change trigger wake-up. The other one is any interrupt in EAL = 1 & function interrupt enable. That's mean users can setup any function with interrupt enable, and the system is waked up until the interrupt issue. Inserting IDLE mode is controlled by idle bit of PCON register. When idle = 1, the system inserts into IDLE mode. After system wake-up from IDLE mode, the idle bit is disabled (zero status) automatically.

- The program stops executing, and full functions are disabled.
- Only the timer with wake-up function is active.
- The oscillator to be the system clock source keeps running, and the other oscillators operation is depend on system operation mode configuration.
- If inserting IDLE mode from normal mode, the system insets to normal mode after wake-up.
- The IDLE mode wake-up sources are P0/P1 level change trigger.
- If the function clock source is system clock, the functions are workable as enabled and under IDLE mode, e.g. Timer, PWM, event counter…
- All interrupt in EAL = 1 & function interrupt enable can wake-up in IDLE mode.

## 9.4 Wake up

Under STOP mode (sleep mode) or idle mode, program doesn't execute. The wakeup trigger can wake the system up to normal mode. The wakeup trigger sources are external trigger (P0/P1 level change) and internal trigger (any interrupt in EAL = 1 & function interrupt enable). The wakeup function builds in interrupt operation issued request flag and trigger system executing interrupt service routine as system wakeup occurrence.

When the system is in STOP mode the high clock oscillator stops. When waked up from STOP mode, MCU waits for 2048 external high-speed oscillator clocks + 5 internal low-speed oscillator clocks and 64 internal high-speed oscillator clocks + 5 internal low-speed oscillator clocks as the wakeup time to stable the oscillator circuit. After the wakeup time, the system goes into the normal mode.

The value of the external high clock oscillator wakeup time is as the following.

The Wakeup time = 1/Fosc * 2048 (sec) + 1/Flosc * 5 + high clock start-up time

Example: In STOP mode (sleep mode), the system is waked up. After the wakeup time, the system goes into normal mode. The wakeup time is as the following.

The wakeup time = 1/Fosc * 2048 + 1/Flosc * 5 = 0.825 ms (Fosc = 4MHz)

The total wakeup time = 0.825 ms + oscillator start-up time

The value of the internal high clock oscillator RC type wakeup time is as the following.

The Wakeup time = 1/Fosc * 64 (sec) + 1/Flosc * 5 + high clock start-up time

Example: In STOP mode (sleep mode), the system is waked up. After the wakeup time, the system goes into normal mode. The wakeup time is as the following.

The wakeup time = 1/Fosc * 64 + 1/Flosc * 5= 315 us (Fhosc = 32MHz)

✷ ***Note: The high clock start-up time is depended on the VDD and oscillator type of high clock.***

Under STOP mode and green mode, the I/O ports with wakeup function are able to wake the system up to normal mode. The wake-up trigger edge is level changing in rising edge or falling edge. The Port 0 and Port 1 have wakeup function. Port 0 wakeup functions always enables, but the Port 1 is controlled by the P1W register.

**P1W Register (0x91)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | P1nW | R/W | 0 | 0: Disable P1.n wakeup functionality<br>1: Enable P1.n wakeup functionality |

# 10 Interrupt

The MCU provides 13 interrupt sources (1 external and 12 interrupt) with 4 priority levels. Each interrupt source includes one or more interrupt request flag(s). When interrupt event occurs, the associated interrupt flag is set to logic 1. If both interrupt enable bit and global interrupt (EAL=1) are enabled, the interrupt request is generated and interrupt service routine (ISR) will be started. Most interrupt request flags must be cleared by software. However, some interrupt request flags can be cleared by hardware automatically. In the end, ISR is finished after complete the RETI instruction. The summary of interrupt source, interrupt vector, priority order and control bit are shown as the table below.

Table 10-1 The interrupt list

| Interrupt | Enable Interrupt | Request (IRQ) | IRQ Clearance | Priority / Vector |
|---|---|---|---|---|
| System Reset | - | - | - | 0 / 0x0000 |
| INT0 | EX0 | IE0 | Automatically | 1 / 0x0003 |
| PWM1 | EPWM1 | PWM1F | By firmware | 2 / 0x0083 |
| I2C | EI2C | SI | By firmware | 3 / 0x0043 |
| Timer 0 | ET0 | TF0 | Automatically | 4 / 0x000B |
| ADC | EADC | ADCF | By firmware | 5 / 0x008B |
| SPI | ESPI | SPIF / MODF | By firmware | 6 / 0x004B |
| T2COM0 | ET2C0 | TF2C0 | Automatically | 7 / 0x0053 |
| Timer 1 | ET1 | TF1 | Automatically | 8 / 0x001B |
| T2COM1 | ET2C1 | TF2C1 | Automatically | 9 / 0x005B |
| UART | ES0 | TI0 / RI0 | By firmware | 10 / 0x0023 |
| T2COM2 | ET2C1 | TF2C2 | Automatically | 11 / 0x0063 |
| Timer 2 | ET2 / ET2RL | TF2 / TF2RL | By firmware | 12 / 0x002B |
| T2COM3 | ET2C3 | TF2C3 | Automatically | 13 / 0x006B |

✷ ***Note: Don't clear* Interrupt request flags *by* firmware *when* Interrupt request flags can be cleared by hardware automatically.**

## 10.1 Interrupt Operation

Interrupt operation is controlled by interrupt request flag and interrupt enable bits. Interrupt request flag is interrupt source event indicator, no matter what interrupt function status (enable or disable). Both interrupt enable bit and global interrupt (EAL=1) are enabled, the system executes interrupt operation when each of interrupt request flags actives. The program counter points to interrupt vector (0x03 – 0x8B) and execute ISR.

## 10.2 Interrupt Priority

Each interrupt source has its specific default priority order. If two interrupts occurs simultaneously, the higher priority ISR will be service first. The lower priority ISR will be serviced after the higher priority ISR completes. The next ISR will be service after the previous ISR complete, no matter the priority order.

For special priority needs, 4-level priority levels (Level 0 – Level 3) are used. All interrupt sources are classified into 6 priority groups (Group0 – Group5). Each group can be set one specific priority level. Priority level is selected by IP0/IP1 registers. Level 3 is the highest priority and Level 0 is the lowest. The interrupt sources inside the same group will share the same priority level. With the same priority level, the priority rule follows default priority.

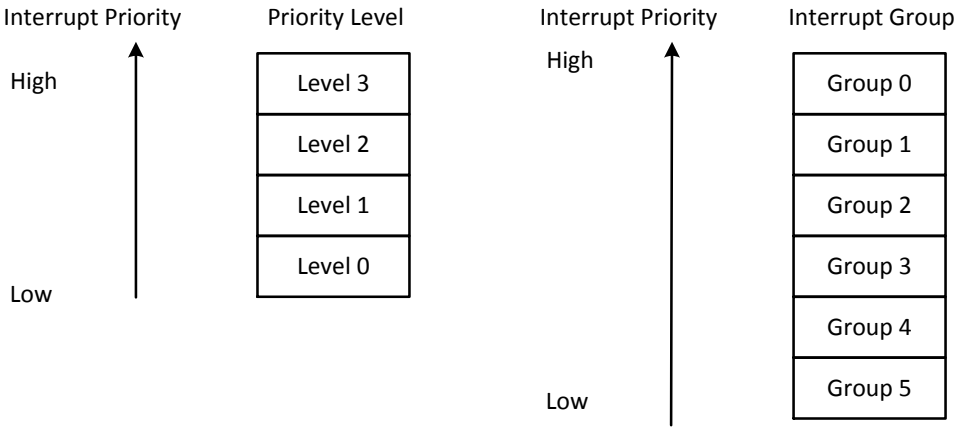
| Priority Level | IP1.x | IP0.x |
|---|---|---|
| Level 0 | 0 | 0 |
| Level 1 | 0 | 1 |
| Level 2 | 1 | 0 |
| Level 3 | 1 | 1 |

The ISR with the higher priority level can be serviced first; even can break the on-going ISR with the lower priority level. The ISR with the lower priority level will be pending until the ISR with the higher priority level completes.

| Group | Interrupt Source | | | |
|---|---|---|---|---|
| Group 0 | INT0 | PWM1 | I2C | - |
| Group 1 | T0 | ADC | SPI | - |
| Group 2 | - | - | T2 COM0 | - |
| Group 3 | T1 | - | T2 COM1 | - |
| Group 4 | UART | - | T2 COM2 | - |
| Group 5 | T2 | - | T2 COM3 | - |

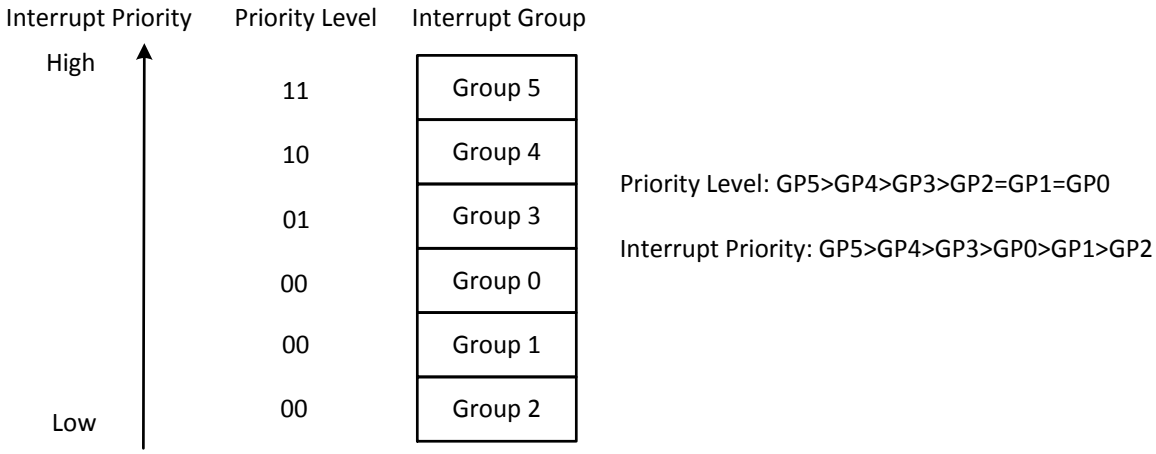
When more than one interrupt request occur, the highest priority request must be executed first. Choose the highest priority request according natural priority and priority level. The steps are as the following:

1. Choose the groups which have the highest priority level between all groups.
2. Choose the group which is the highest nature priority between the groups with the highest priority level.
3. Choose the ISR which has the highest nature priority inside the group with the highest priority.




As the example, group5 has the highest priority level and group0~group2 have the lowest priority level. It means the interrupt vector in group5 has the highest interrupt priority, the 2nd interrupt priority in group4 and the 3rd interrupt priority in group3. Group0~ group2 have the same priority level thus the nature priority rule will be followed. Therefore, interrupt priority will be group5> group4> group3> group0> group1> group2.

```
MOV     IP0, #00101000B         ; Set group0 - group5 in different priority level.
MOV     IP1, #00110000B
```



**IP0, IP1 Registers**

| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| IP0 | - | - | IP05 | IP04 | IP03 | IP02 | IP01 | IP00 |
| IP1 | - | - | IP15 | IP14 | IP13 | IP12 | IP11 | IP10 |

**IP0 Register (0XA9)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 5..0 | IP0[5:0] | R/W | 0 | Interrupt priority. Each bit together with corresponding bit from IP1 register specifies the priority level of the respective interrupt priority group. |
| Else | Reserved | R | 0 | |

**IP1 Register (0XB9)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 5..0 | IP1[5:0] | R/W | 0 | Interrupt priority. Each bit together with corresponding bit from IP0 register specifies the priority level of the respective interrupt priority group. |
| Else | Reserved | R | 0 | |

## 10.3 Interrupt Registers

| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| IEN0 | EAL | - | ET2 | ES0 | ET1 | - | ET0 | EX0 |
| IEN1 | ET2RL | - | ET2C3 | ET2C2 | ET2C1 | ET2C0 | ESPI | EI2C |
| IEN2 | - | - | - | - | - | - | EADC | - |
| IEN4 | EPWM1 | - | - | - | PWM1F | - | - | - |
| IRCON | TF2RL | TF2 | TF2C3 | TF2C2 | TF2C1 | TF2C0 | - | - |
| IRCON2 | - | - | - | | - | - | - | ADCF |
| TCON | TF1 | TR1 | TF0 | TR0 | - | - | IE0 | - |
| S0CON | SM0 | SM1 | SM20 | REN0 | TB80 | RB80 | TI0 | RI0 |
| SPSTA | SPIF | WCOL | SSERR | MODF | - | - | - | - |
| I2CCON | CR2 | ENS1 | STA | STO | SI | AA | CR1 | CR0 |

**IEN0 Register (0XA8)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | EAL | R/W | 0 | Enable all interrupt control bit.<br>0: Disable all interrupt function.<br>1: Enable all interrupt function. |
| 5 | ET2 | R/W | 0 | T2 timer interrupt control bit<br>0: Disable T2 interrupt function.<br>1: Enable T2 interrupt function. |
| 4 | ES0 | R/W | 0 | UART interrupt control bit.<br>0: Disable UART interrupt function.<br>1: Enable UART interrupt function. |
| 3 | ET1 | R/W | 0 | T1 timer interrupt control bit.<br>0: Disable T1 interrupt function.<br>1: Enable T1 interrupt function. |
| 1 | ET0 | R/W | 0 | T0 timer interrupt control bit.<br>0: Disable T0 interrupt function.<br>1: Enable T0 interrupt function |
| 0 | EX0 | R/W | 0 | External P1.0 interrupt (INT0) control bit.<br>0: Disable INT0 interrupt function.<br>1: Enable INT0 interrupt function. |
| Else | Reserved | R | 0 | |

**IEN1 Register (0XB8)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | ET2RL | R/W | 0 | T2 Timer external reload interrupt control bit.<br>0: Disable T2 external reload interrupt function.<br>1: Enable T2 external reload interrupt function. |
| 5 | ET2C3 | R/W | 0 | T2 Timer COM3 interrupt control bit.<br>0: Disable T2COM3 interrupt function.<br>1: Enable T2COM3 interrupt function. |
| 4 | ET2C2 | R/W | 0 | T2 Timer COM2 interrupt control bit.<br>0: Disable T2COM2 interrupt function.<br>1: Enable T2COM2 interrupt function. |
| 3 | ET2C1 | R/W | 0 | T2 Timer COM1 interrupt control bit.<br>0: Disable T2COM1 interrupt function.<br>1: Enable T2COM1 interrupt function. |
| 2 | ET2C0 | R/W | 0 | T2 Timer COM0 interrupt control bit.<br>0: Disable T2COM0 interrupt function.<br>1: Enable T2COM0 interrupt function. |
| 1 | ESPI | R/W | 0 | SPI interrupt control bit<br>0: Disable SPI interrupt function.<br>1: Enable SPI interrupt function. |
| 0 | EI2C | R/W | 0 | I2C interrupt control bit.<br>0: Disable I2C interrupt function.<br>1: Enable I2C interrupt function. |
| Else | Reserved | R | 0 | |

**IEN2 Register (0X9A)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 1 | EADC | R/W | 0 | ADC interrupt control bit.<br>0: Disable ADC interrupt function.<br>1: Enable ADC interrupt function. |
| Else | Reserved | R | 0 | |

**IEN4 Register (0XD1)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | EPWM1 | R/W | 0 | PWM1 interrupt control bit.<br>0 = Disable PWM1 interrupt function.<br>1 = Enable PWM1 interrupt function. |
| 3 | PWM1F | R/W | 0 | PWM1 interrupt request flag.<br>0: None PWM1 interrupt request<br>1: PWM1 interrupt request. |
| Else | Reserved | R | 0 | |

**IRCON Register (0xC0)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | TF2RL | R/W | 0 | T2 timer external reload interrupt request flag.<br>0: None TF2RL interrupt request<br>1: TF2RL interrupt request. |
| 6 | TF2 | R/W | 0 | T2 timer interrupt request flag.<br>0: None T2 interrupt request.<br>1: T2 interrupt request. |
| 5 | TF2C3 | R/W | 0 | T2 Timer COM3 interrupt request flag.<br>0: None T2COM3 interrupt request.<br>1: T2COM3 interrupt request. |
| 4 | TF2C2 | R/W | 0 | T2 Timer COM2 interrupt request flag.<br>0: None T2COM2 interrupt request.<br>1: T2COM2 interrupt request. |
| 3 | TF2C1 | R/W | 0 | T2 Timer COM1 interrupt request flag.<br>0: None T2COM1 interrupt request.<br>1: T2COM1 interrupt request. |
| 2 | TF2C0 | R/W | 0 | T2 Timer COM0 interrupt request flag.<br>0: None T2COM0 interrupt request.<br>1: T2COM0 interrupt request. |
| Else | Reserved | R | 0 | |

**IRCON2 Register (0XBF)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 0 | ADCF | R/W | 0 | ADC interrupt request flag.<br>0: None ADC interrupt request.<br>1: ADC interrupt request. |
| Else | Reserved | R | 0 | |

**TCON Register (0X88)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | TF1 | R/W | 0 | T1 timer external reload interrupt request flag.<br>0: None T1 interrupt request<br>1: T1 interrupt request. |
| 5 | TF0 | R/W | 0 | T0 timer external reload interrupt request flag.<br>0: None T0 interrupt request<br>1: T0 interrupt request. |
| 1 | IE0 | R | 0 | External P1.0 interrupt (INT0) request flag<br>0: None INT0 interrupt request.<br>1: INT0 interrupt request. |
| Else | | | | Refer to other chapter(s) |

**S0CON Register (0X98)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 1 | TI0 | R/W | 0 | UART transmit interrupt request flag. It indicates completion of a serial transmission at UART. It is set by hardware at the end of bit 8 in mode 0 or at the beginning of a stop bit in other modes. It must be cleared by software.<br>0: None UART transmit interrupt request.<br>1: UART transmit interrupt request. |
| 0 | RI0 | R/W | 0 | UART receive interrupt request flag. It is set by hardware after completion of a serial reception at UART. It is set by hardware at the end of bit 8 in mode 0 or in the middle of a stop bit in other modes. It must be cleared by software.<br>0: None UART receive interrupt request.<br>1: UART receive interrupt request. |
| Else | | | | Refer to other chapter(s) |

**SPSTA Register (0XE1)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | SPIF | R | 0 | SPI complete communication flag<br>Set automatically at the end of communication<br>Cleared automatically by reading SPSTA, SPDAT registers |
| 4 | MODF | R | 0 | Mode fault flag |
| Else | | | | Refer to other chapter(s) |

**I2CCON Register (0XDC)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | SI | R/W | 0 | Serial interrupt flag<br>The SI is set by hardware when one of 25 out of 26 possible I2C states is entered. The only state that does not set the SI is state F8h, which indicates that no relevant state information is available. The SI flag must be cleared by software. In order to clear the SI bit, '0' must be written to this bit. Writing a '1' to SI bit does not change value of the SI. |
| Else | | | | Refer to other chapter(s) |

## 10.4 Example

Defining Interrupt Vector. The interrupt service routine is following user assembly code program.

```
                ORG     0               ; 0000H
                JMP     START           ; Jump to user program address.
                …
                ORG     0X000B          ; Jump to interrupt service routine address.
                JMP     ISR_T0
                ORG     0X001B
                JMP     ISR_T1
                …
                ORG     0X008B
                JMP     ISR_ADC
                …
                ORG     0X00EC
START:                                  ; 00ECH, The head of user program.
                …                       ; User program.
                …
                JMP     START           ; End of user program.
                …
ISR_T0:                                 ; The head of interrupt service routine.
                PUSH    ACC             ; Save ACC to stack buffer.
                PUSH    PSW             ; Save PSW to stack buffer.
                …
                POP     PSW             ; Load PSW from stack buffer.
                POP     ACC             ; Load ACC from stack buffer.
                RETI                    ; End of interrupt service routine.
ISR_ADC:                                ;
                PUSH    ACC             ; Save ACC to stack buffer.
                PUSH    PSW             ; Save PSW to stack buffer.
                …
                POP     PSW             ; Load PSW from stack buffer.
                POP     ACC             ; Load ACC from stack buffer.
                RETI                    ; End of interrupt service routine.
                …
ISR_T1                                  ;
                PUSH    ACC             ; Save ACC to stack buffer.
                PUSH    PSW             ; Save PSW to stack buffer.
                …
                POP     PSW             ; Load PSW from stack buffer.
                POP     ACC             ; Load ACC from stack buffer.
                RETI                    ; End of interrupt service routine.

                END                     ; End of program.
```

# 11 GPIO

The microcontroller has up to 18 bidirectional general purpose I/O pin (GPIO). Unlike the original 8051 only has open-drain output, SN8F5702 builds in push-pull output structure to improve its driving performance.

## 11.1 Input and Output Control

The input and output direction control is configurable through P0M to P2M registers. These bits specify each pin that is either input mode or output mode.

| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| P0M | P07M | P06M | P05M | P04M | P03M | P02M | P01M | P00M |
| P1M | P17M | P16M | P15M | P14M | P13M | P12M | P11M | P10M |
| P2M | - | - | - | - | - | - | P21M | P20M |
| P0OC | - | - | - | P15OC | P14OC | P13OC | P06OC | P05OC |

**P0M: 0xF9, P1M: 0xFA, P2M: 0xFB**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | P07M | R/W | 0 | Mode selection of P0.7<br>0: Input mode<br>1: Output mode |
| 6 | P06M | R/W | 0 | Mode selection of P0.6<br>0: Input mode<br>1: Output mode |
| 5 | P05M | R/W | 0 | Mode selection of P0.5<br>0: Input mode<br>1: Output mode |
| 4..0 | | | | et cetera |

**P0OC Register (0xE4)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..5 | | R/W | 000 | Refer to PWM chapter |
| 4 | P15OC | R/W | 0 | P1.5 open-drain output mode<br>0: Disable<br>1: Enable, output high status becomes to input mode |
| 3 | P14OC | R/W | 0 | P1.4 open-drain output mode<br>0: Disable<br>1: Enable, output high status becomes to input mode |
| 2 | P13OC | R/W | 0 | P1.3 open-drain output mode<br>0: Disable<br>1: Enable, output high status becomes to input mode |
| 1 | P06OC | R/W | 0 | P0.6 open-drain output mode<br>0: Disable<br>1: Enable, output high status becomes to input mode |
| 0 | P05OC | R/W | 0 | P0.5 open-drain output mode<br>0: Disable<br>1: Enable, output high status becomes to input mode |

## 11.2 Input Data and Output Data

By a read operation from any registers of P0 to P2, the current pin's logic level would be fetch to represent its external status. This operation remains functional even the pin is shared with other function like UART and I2C which can monitor the bus condition in some case.

A write P0 to P2 register value would be latched immediately, yet the value would be outputted until the mapped P0M – P2M is set to output mode. If the pin is currently in output mode, any value set to P0 to P2 register would be presented on the pin immediately.

| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| P0 | P07 | P06 | P05 | P04 | P03 | P02 | P01 | P00 |
| P1 | P17 | P16 | P15 | P14 | P13 | P12 | P11 | P10 |
| P2 | - | - | - | - | - | - | P21 | P20 |

**P0: 0x80, P1: 0x90, P2: 0xA0**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | P07 | R/W | 1 | Read: P0.7 pin's logic level<br>Write 1/0: Output logic high or low (applied if P07M = 1) |
| 6 | P06 | R/W | 1 | Read: P0.6 pin's logic level<br>Write 1/0: Output logic high or low (applied if P06M = 1) |
| 5 | P05 | R/W | 1 | Read: P0.5 pin's logic level<br>Write 1/0: Output logic high or low (applied if P05M = 1) |
| 4..0 | | | | et cetera |

## 11.3 On-chip Pull-up Resisters

The P0UR to P2UR registers are mapped to each pins' internal 100 kΩ (in typical value) pull-up resister.

| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| P0UR | P07UR | P06UR | P05UR | P04UR | P03UR | P02UR | P01UR | P00UR |
| P1UR | P17UR | P16UR | P15UR | P14UR | P13UR | P12UR | P11UR | P10UR |
| P2UR | - | - | - | - | - | - | P21UR | P20UR |

**P0UR: 0xF1, P1UR: 0xF2, P2UR: 0xF3**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | P07UR | R/W | 0 | On-chip pull-up resister control of P0.7<br>0: Disable*<br>1: Enable |
| 6 | P06UR | R/W | 0 | On-chip pull-up resister control of P0.6<br>0: Disable*<br>1: Enable |
| 5 | P05UR | R/W | 0 | On-chip pull-up resister control of P0.5<br>0: Disable*<br>1: Enable |
| 4..0 | | | | et cetera |

* Recommended disable pull-up resister if the pin is output mode or analog function

## 11.4 Pin Shared with Analog Function

The microcontroller builds in analog functions, such as ADC. The Schmitt trigger of input channel is strongly recommended to switch off if the pin's shared analog function is enabled.

| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| P0CON | P0CON7 | - | - | - | - | - | - | - |
| P1CON | P1CON7 | P1CON6 | P1CON5 | P1CON4 | P1CON3 | P1CON2 | P1CON1 | P1CON0 |
| P2CON | - | - | - | - | - | - | P2CON1 | P2CON0 |

**P2CON: 0x9E, P1CON: 0xD6, P0CON: 0x9D**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | P1CON7 | R/W | 0 | Schmitt trigger control of P1.7<br>0: Enable<br>1: Disable |
| 6 | P1CON6 | R/W | 0 | Schmitt trigger control of P1.6<br>0: Enable<br>1: Disable |
| 5 | P1CON5 | R/W | 0 | Schmitt trigger control of P1.5<br>0: Enable<br>1: Disable |
| 4..0 | | | | et cetera |

# 12 External Interrupt

INT0 is external interrupt trigger sources. Build in edge trigger configuration function and edge direction is selected by PEDGE register. When both external interrupt (EX0) and global interrupt (EAL) are enabled, the external interrupt request flag (IE0) will be set to "1" as edge trigger event occurs. The program counter will jump to the interrupt vector (ORG 0x0003) and execute interrupt service routine. Interrupt request flag will be cleared by hardware before ISR is executed.

## 12.1 External Interrupt Registers

| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PEDGE | - | - | - | - | - | - | EX0G1 | EX0G0 |
| IEN0 | EAL | - | ET2 | ES0 | ET1 | - | ET0 | EX0 |
| TCON | TF1 | TR1 | TF0 | TR0 | - | - | IE0 | - |

**PEDGE Register (0X8F)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 1..0 | EX0G[1:0] | R/W | 10 | External interrupt 0 trigger edge control register.<br>00: Reserved.<br>01: Rising edge trigger.<br>10: Falling edge trigger (default)<br>11: Both rising and falling edge trigger |
| Else | Reserved | R | 0 | |

**IEN0 Register (0XA8)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | EAL | R/W | 0 | Enable all interrupt control bit.<br>0: Disable all interrupt function.<br>1: Enable all interrupt function. |
| 0 | EX0 | R/W | 0 | External P1.0 interrupt (INT0) control bit.<br>0: Disable INT0 interrupt function.<br>1: Enable INT0 interrupt function. |
| Else | | | | Refer to other chapter(s) |

**TCON Register (0X88)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 1 | IE0 | R/W | 0 | External P1.0 interrupt (INT0) request flag<br>0: None INT0 interrupt request.<br>1: INT0 interrupt request. |
| Else | | | | Refer to other chapter(s) |

✷ ***Note: Before* clear one of *TF0, TF1 or IE0 flag* manually by firmware, *user must be made sure others request flag in TCON register doesn't active.***

## 12.2 Sample Code

The following sample code demonstrates how to perform INT0 with interrupt.

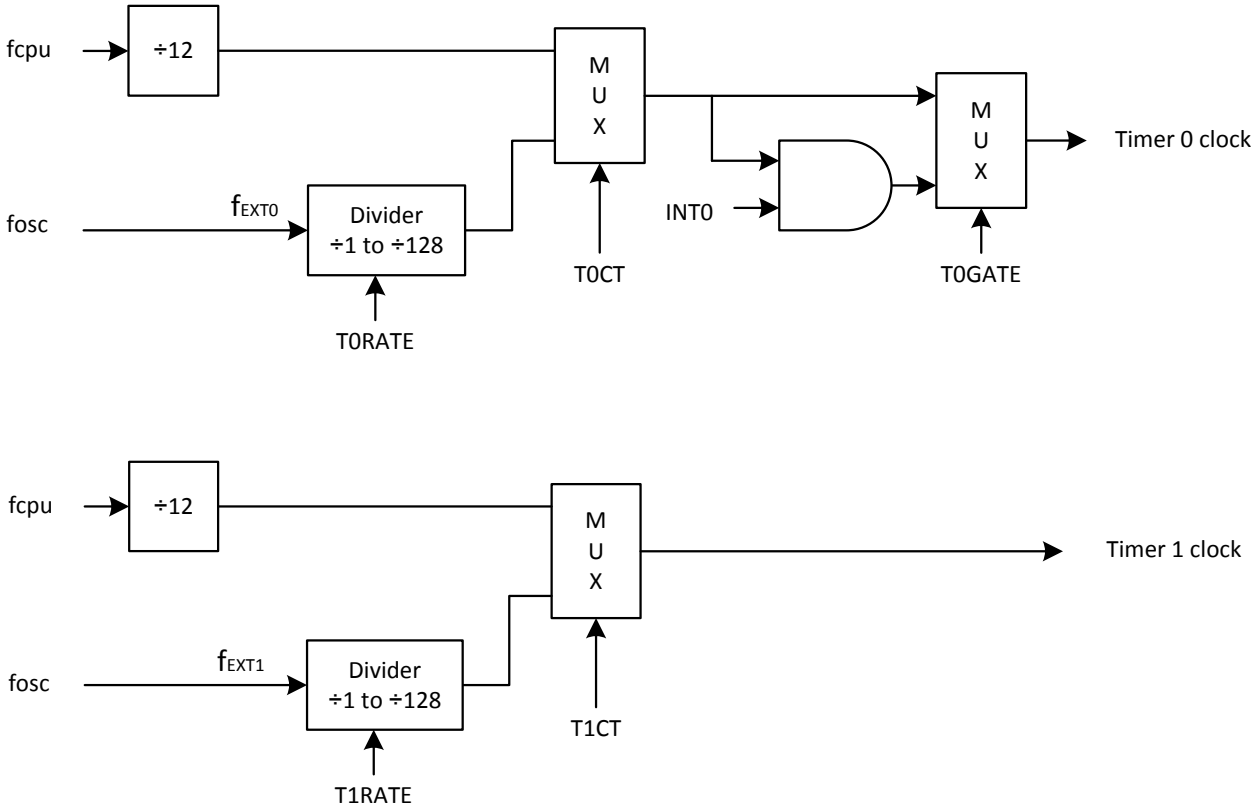
```
#define INT0Rsing      (1 << 0)  //INT0 trigger edge is rising edge
#define INT0Falling    (2 << 0)  //INT0 trigger edge is falling edge
#define INT0LeChge     (3 << 0)  //INT0 trigger edge is level chagne
#define EINT0          (1 << 0)  //INT0 interrupt enable

void EnableINT(void)
{
  // INT0 rising edge
  PEDGE = INT0Rising;

  // Enable INT0 interrupt
  IEN0 |= EINT0;
  // Enable total interrupt
  IEN0 |= 0x80;

  P0 = 0x00;
  P0M = 0x01;
}

void INT0Interrupt(void) interrupt ISRInt0  //0x03
{ //IE0 clear by hardware
  P00 = ~P00;
}

```

# 13 Timer 0 and Timer 1

Timer 0 and Timer 1 are two independent binary up timers. Timer 0 has four different operation modes: (1) 13-bit up counting timer, (2) 16-bit up counting timer, (3) 8-bit up counting timer with specified reload value support, and (4) separated two 8-bit up counting timer. By contrast, Timer 1 has only mode 0 to mode 2 which are same as Timer 0. Timer 0 and Timer 1 respectively support ET0 and ET1 interrupt function.

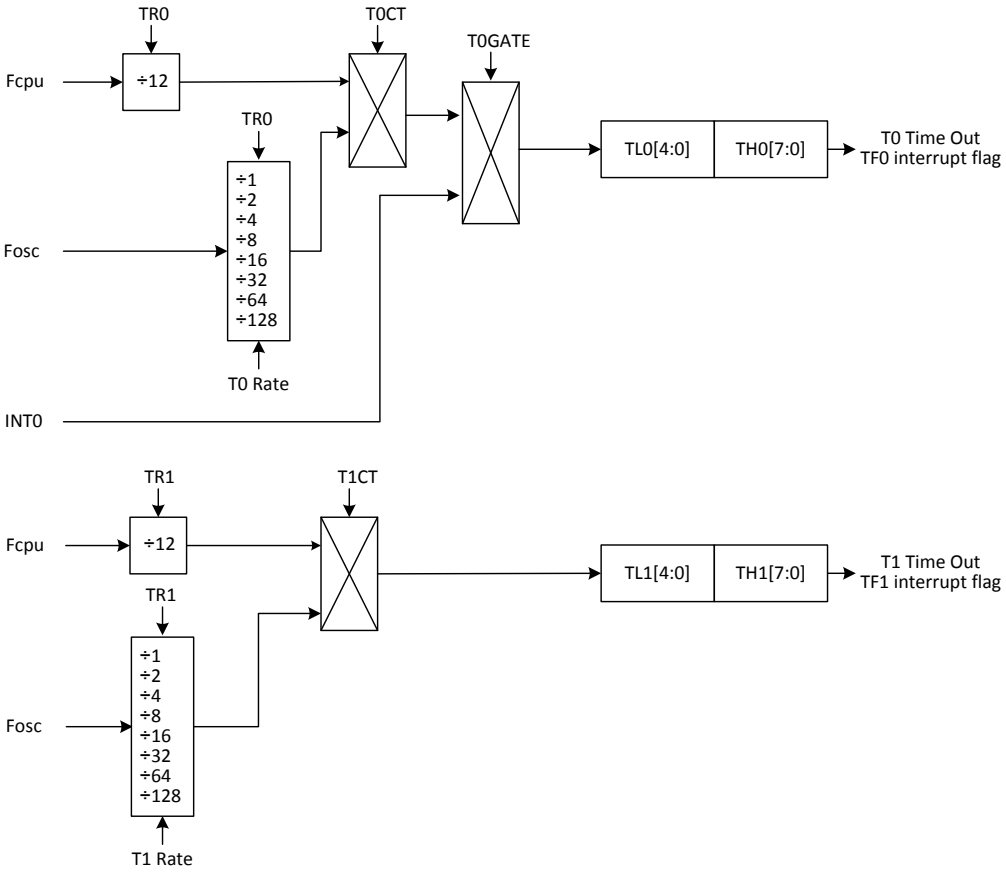
## 13.1 Timer 0 and Timer 1 Clock Selection

The figures below illustrate the clock selection circuit of Timer 0 and Timer 1. Timer 0 has two clock sources selection: fcpu and fosc. All clock sources can be gated (pause) by INT0 pin if T0GATE is applied. Timer 1 clock sources selection: fcpu and fosc.



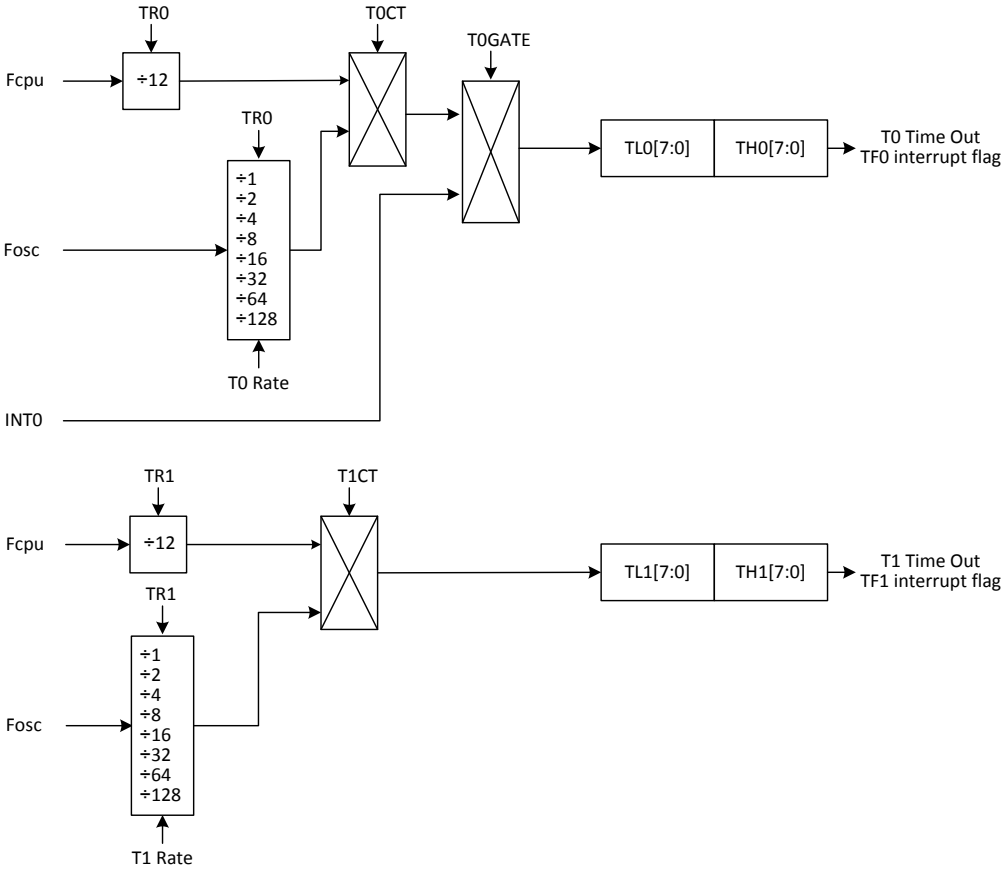
## 13.2 Mode 0: 13-bit Up Counting Timer

Timer 0 and Timer 1 in mode 0 is a 13-bit up counting timer (the upper 3 bits of TL0 is suspended). Once the timer's counter is overflow (counts from 0xFF1F to 0x0000), TF0/TF1 flag would be issued immediately. This flag is readable and writable by firmware if ET0/ET1 does not apply, or can be handled by interrupt controller if ET0/ET1 is applied.



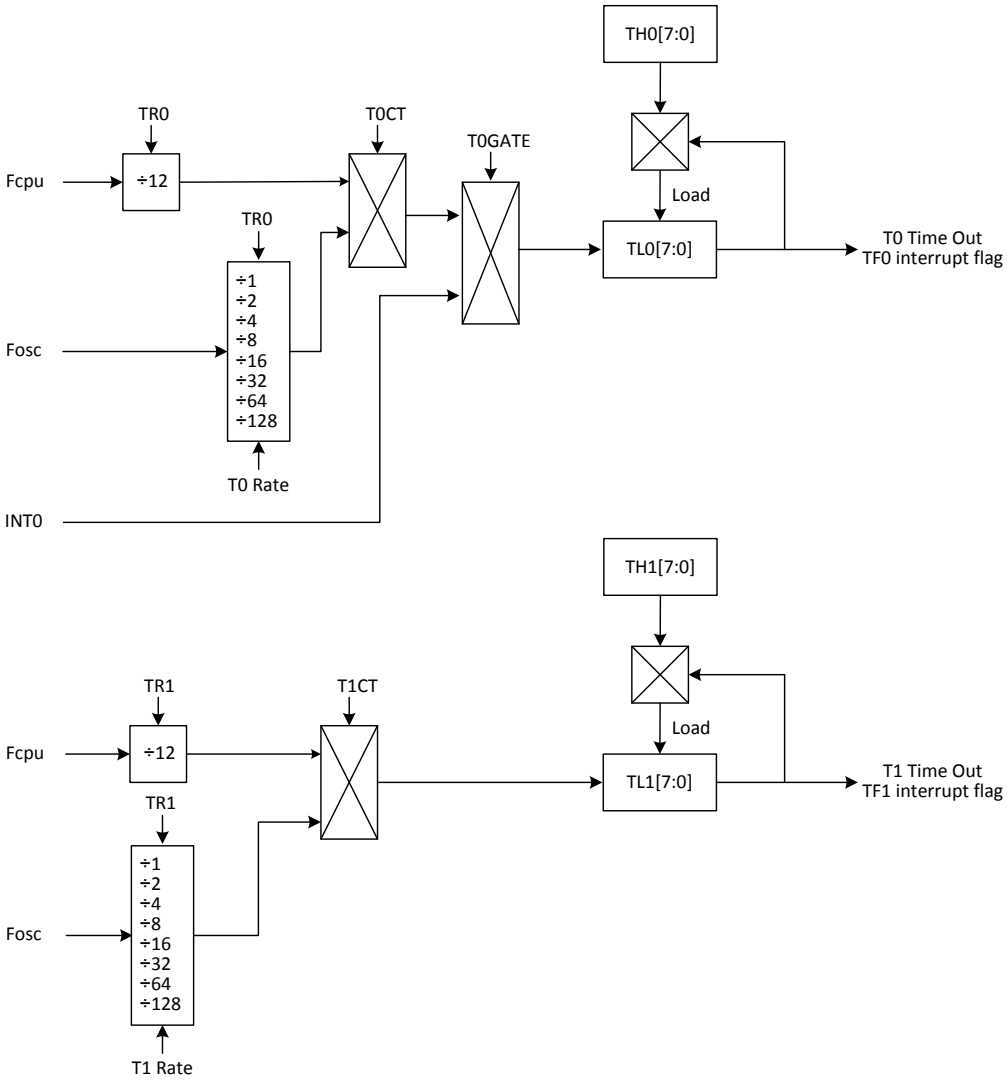
## 13.3 Mode 1: 16-bit Up Counting Timer

Timer 0 and Timer 1 in mode 1 is a 16-bit up counting timer. Once the timer's counter overflow is occurred (from 0xFFFF to 0x0000), TF0/TF1 would be issued which is readable by firmware or can be handled by interrupt controller (if ET0/ET1 applied).



## 13.4 Mode 2: 8-bit Up Counting Timer with Specified Reload Value Support

Timer 0 and Timer 1 in mode 2 is an 8-bit up counting timer (TL0/TL1) with a specifiable reload value. An overflow event (TL0/TL1 counts from 0xFF to 0x00) issues its TF0/TF1 flag for firmware or interrupt controller; meanwhile, the timer duplicates TH0/TH1 value to TL0/TL1 register in the same time. As a result, the timer is actually counts from 0xFF to the value of TH0/TH1.

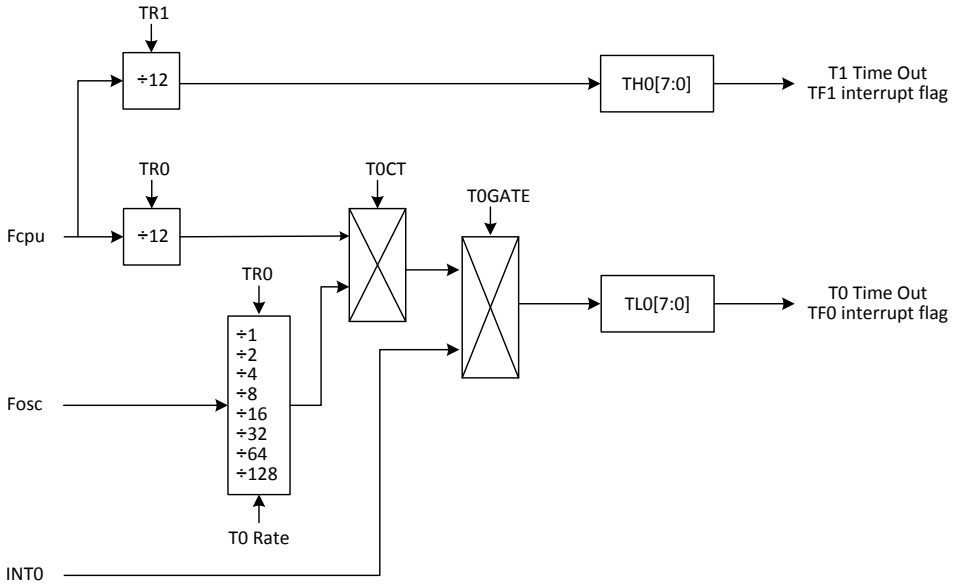


## 13.5 Mode 3 (Timer 0 only): Separated Two 8-bit Up Counting Timer

Mode 3 treats TH0 and TL0 as two separated 8-bit timers. TL0 is an 8-bit up counting timer with two clock sources selection (fcpu and fosc), whereas TH0 clock source is fixed at fcpu/12. Only TL0 clock source can be gated (pause) by INT0 pin if T0GATE is applied.

In this mode TL0 counter is enabled by TR0, and its overflow signal is reflected in TF0 flag. TH0 counter is controlled by TR1, and TF1 flag is also occupied by TH0 overflow signal.

Timer 1 cannot issue any overflow event in this situation, and it can be seen as a self-counting timer without flag support.



## 13.6 Timer 0 and Timer 1 Registers

| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TCON | TF1 | TR1 | TF0 | TR0 | - | - | IE0 | - |
| TCON0 | - | T0RATE2 | T0RATE1 | T0RATE0 | - | T1RATE2 | T1RATE1 | T1RATE0 |
| TMOD | - | T1CT | T1M1 | T1M0 | T0GATE | T0CT | T0M1 | T0M0 |
| TH0 | TH07 | TH06 | TH05 | TH04 | TH03 | TH02 | TH01 | TH00 |
| TL0 | TL07 | TL06 | TL05 | TL04 | TL03 | TL02 | TL01 | TL00 |
| TH1 | TH17 | TH16 | TH15 | TH14 | TH13 | TH12 | TH11 | TH10 |
| TL1 | TL17 | TL16 | TL15 | TL14 | TL13 | TL12 | TL11 | TL10 |
| IEN0 | EAL | - | ET2 | ES0 | ET1 | - | ET0 | EX0 |

**IEN0 Register (0xA8)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | EAL | R/W | 0 | Interrupts enable. Refer to Chapter Interrupt |
| 3 | ET1 | R/W | 0 | Timer 1 interrupt<br>0: Disable<br>1: Enable |
| 1 | ET0 | R/W | 0 | Timer 0 interrupt<br>0: Disable<br>1: Enable |

**TH0 / TH1 Registers (TH0: 0x8C, TH1: 0x8D)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | TH0/TH1 | R/W | 0x00 | High byte of Timer 0 and Timer 1 counter |

**TL0 / TL1 Register (TL0: 0x8A, TL1: 0x8B)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | TL0/TL1 | R/W | 0x00 | Low byte of Timer 0 and Timer 1 counter |

**TCON Register (0x88)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | TF1 | R/W | 0 | Timer 1 overflow event<br>0: Timer 1 does not have any overflow event<br>1: Timer 1 has overflowed<br>This bit can be cleared automatically by interrupt handler, or manually by firmware |
| 6 | TR1 | R/W | 0 | Timer 1 function<br>0: Disable<br>1: Enable |
| 5 | TF0 | R/W | 0 | Timer 0 overflow event<br>0: Timer 0 does not have any overflow event<br>1: Timer 0 has overflowed<br>This bit can be cleared automatically by interrupt handler, or manually by firmware |
| 4 | TR0 | R/W | 0 | Timer 0 function<br>0: Disable<br>1: Enable |
| 3..2 | Reserved | R | 0 | |
| 1 | IE0 | R/W | 0 | Refer to INT0 |
| 0 | Reserved | R | 0 | |

✷ ***Note: Before* clear one of *TF0, TF1 or IE0 flag* manually by firmware, *user must be made sure others request flag in TCON register doesn't active.***

**TCON0 Register (0xE7)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | Reserved | R | 0 | |
| 6..4 | T0RATE[2:0] | R/W | 000 | Clock divider of Timer 0 external clock source *(1)<br>000: $f_{EXT0}$ / 128<br>001: $f_{EXT0}$ / 64<br>010: $f_{EXT0}$ / 32<br>011: $f_{EXT0}$ / 16<br>100: $f_{EXT0}$ / 8<br>101: $f_{EXT0}$ / 4<br>110: $f_{EXT0}$ / 2<br>111: $f_{EXT0}$ / 1 |
| 3 | Reserved | R | 0 | |
| 2..0 | T1RATE[2:0] | R/W | 000 | Clock divider of Timer 0 external clock source *(2)<br>000: $f_{EXT1}$ / 128<br>001: $f_{EXT1}$ / 64<br>010: $f_{EXT1}$ / 32<br>011: $f_{EXT1}$ / 16<br>100: $f_{EXT1}$ / 8<br>101: $f_{EXT1}$ / 4<br>110: $f_{EXT1}$ / 2<br>111: $f_{EXT1}$ / 1 |

*(1) $f_{EXT1}$ = fosc.

*(2) $f_{EXT0}$ = fosc.

**TMOD Register (0x89)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | Reserved | R | 0 | |
| 6 | T1CT | R/W | 0 | Timer 1 clock source selection<br>0: $f_{Timer1}$ = fcpu / 12<br>1: $f_{Timer\ 1}$ = $f_{EXT1}$ / T1RATE (refer to T1RATE) *(1) |
| 5..4 | T1M[1:0] | R/W | 00 | Timer 1 operation mode<br>00: 13-bit up counting timer<br>01: 16-bit up counting timer<br>10: 8-bit up counting timer with reload support<br>11: Reserved |
| 3 | T0GATE | R/W | 0 | Timer 0 gate control mode |

| | | | | |
|---|---|---|---|---|
| | | | | 0: Disable<br>1: Enable, Timer 0 clock source is gated by INT0 |
| 2 | T0CT | R/W | 0 | Timer 0 clock source selection<br>0: $f_{Timer0}$ = fcpu / 12<br>1: $f_{Timer0}$ = fEXT0 / T0RATE (refer to T0RATE) *(2) |
| 1..0 | T0M[1:0] | R/W | 00 | Timer 0 operation mode<br>00: 13-bit up counting timer<br>01: 16-bit up counting timer<br>10: 8-bit up counting timer with reload support<br>11: Separated two 8-bit up counting timer |

*(1) fEXT1 = fosc.

*(2) fEXT0 = fosc.

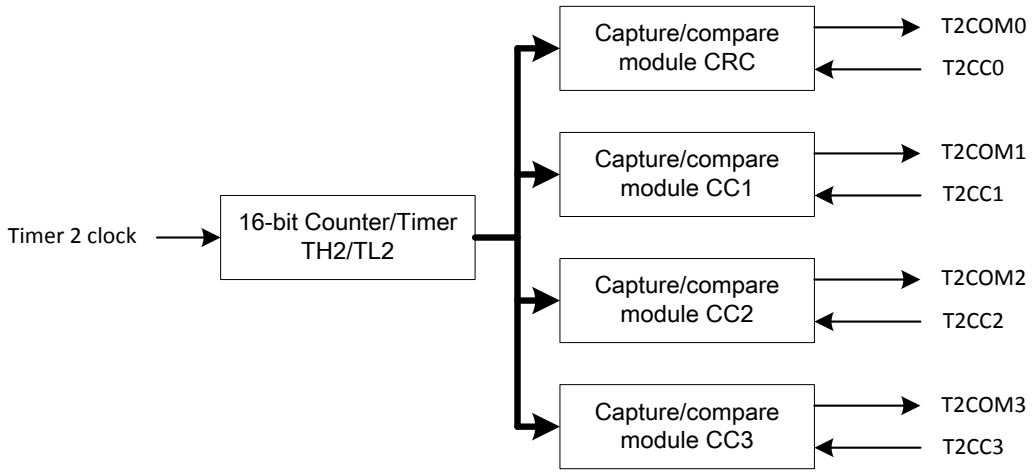
## 13.7 Sample Code

The following sample code demonstrates how to perform T0/T1 with interrupt.

```
#define T0Mode0      (0 << 0)  //T0 mode0, 13-bit counter
#define T0Mode1      (1 << 0)  //T0 mode1, 16-bit counter
#define T0Mode2      (2 << 0)  //T0 mode2, 8-bit auto-reload counter
#define T0Mode3      (3 << 0)  //T0 mode3, T0 two 8-bit counter/T1 no flag
#define T0GATE       (8 << 0)  //T0 gating clock by INT0
#define T0ClkFcpu    (0 << 0)  //T0 clock source from Fcpu/12
#define T0ClkExt     (4 << 0)  //T0 clock source from Fosc
#define T0ExtFosc    (0 << 4)  //T0 clock source from Fosc

#define T1Mode0      (0 << 4)  //T1 mode0, 13-bit counter
#define T1Mode1      (1 << 4)  //T1 mode1, 16-bit counter
#define T1Mode2      (2 << 4)  //T1 mode2, 8-bit auto-reload counter
#define T1Mode3      (3 << 4)  //T1 mode3, T1 stop
#define T1ClkFcpu    (0 << 4)  //T0 clock source from Fcpu/12
#define T1ClkExt     (4 << 4)  //T0 clock source from Fosc

void InitT0T1(void)
{
  // T0/T1_Initial
  TH0 = 0x00;
  TL0 = 0x00;
  TH1 = 0x00;
  TL1 = 0x00;
  // T0 mode0 with gating clock by INT0, clock source from Fosc
  TMOD |= T0Mode0 | T0GATE | T0ClkExT;
  // T0 clock source = Fosc/1
  TCON0 |= 0x70;
  // T1 mode1, clock source from Fcpu/12
  TMOD |= T1Mode1 | T1ClkFcpu;
  // Timer 0/1 enable. Clear TF0/TF1
  TCON |= 0x50;
  // Enable T0/T1 interrupt
  IEN0 |= 0x0A;
  // Enable total interrupt
  IEN0 |= 0x80;

  P0 = 0x00;
  P0M = 0x03;
}

void T0Interrupt(void) interrupt ISRTimer0  //0x0B
{ //TF0 clear by hardware
  P00 = ~P00;
}
void T1Interrupt(void) interrupt ISRTimer1  //0x1B
{ //TF1 clear by hardware
  P01 = ~P01;
}
```

# 14 Timer 2

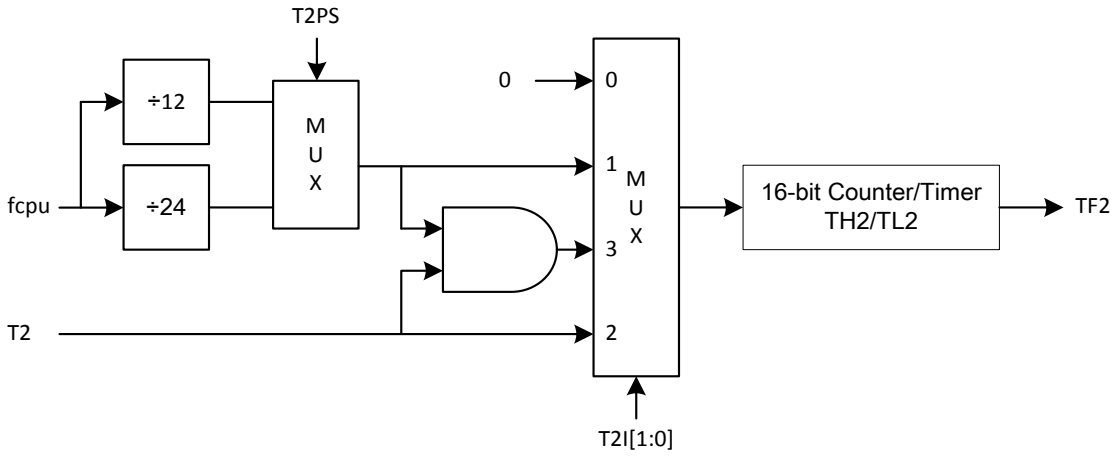
Timer 2 is a 16-bit up counting timer which has several optional extensions: specified reload value, comparison output (PWM) and capture function. Timer 2 consists of a dedicated 16-bit counter/timer and four 16-bit capture/compare modules. Each capture/compare module has its own associated I/O when enabled. Each capture/compare module may be configured to operate independently in one of 3 modes: compare, capture with rising edge, or capture with register be written.




## 14.1 Timer 2 Up-counting Control

Timer 2 has three operation modes by its clock source: specify fcpu clocks (fcpu/12 and fcpu/24), specify fcpu clocks with a stop control, and external clock input. The table below categorizes these three operation modes and its related registers (T2I1, T2I0 and T2PS). Once the timer's counter is overflow (counts from 0xFFFF to 0x0000), TF2 would be issued immediately which can read/write by firmware. Timer 2 interrupt function is controlled by ET2.

| T2I1 | T2I0 | T2PS | Timer 2 Clock Source |
|---|---|---|---|
| 0 | 0 | X | Disable Timer 2 |
| 0 | 1 | 0 | fcpu/12 |
| 0 | 1 | 1 | fcpu/24 |
| 1 | 1 | 0 | fcpu/12 (stop counting when T2 pin is low, resume when T2 is high) |
| 1 | 1 | 1 | fcpu/24 (stop counting when T2 pin is low, resume when T2 is high) |
| 1 | 0 | X | T2 pin rising edge (T2 pin clock rate ≤ 0.5 * fcpu) |

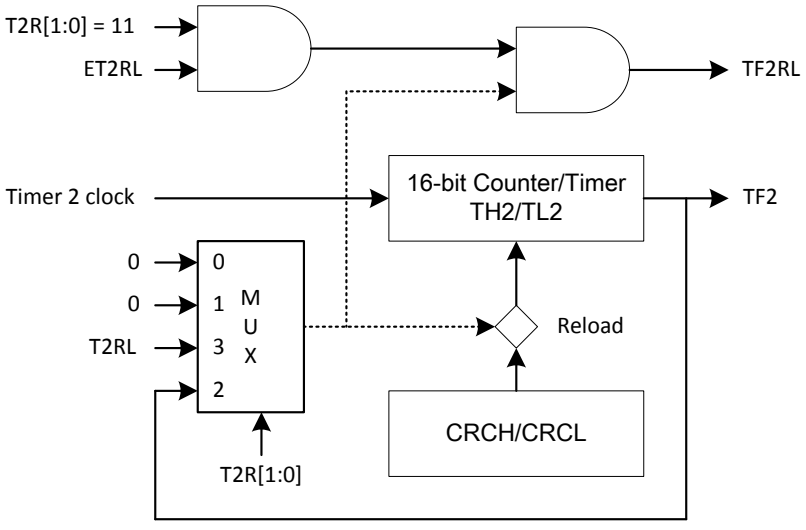



## 14.2 Specified Timer 2 Reload Value

The specified reload value is an optional function which can reload Timer 2 counter by overflow or external control pin.

If overflow-to-reload is selected, Timer 2 duplicates CRCH/CRCL value to its counter (TH2/TL2) automatically by overflow signal. As a result, Timer 2 would repeatedly counts from CRCH/CRCL value to 0xFFFF.

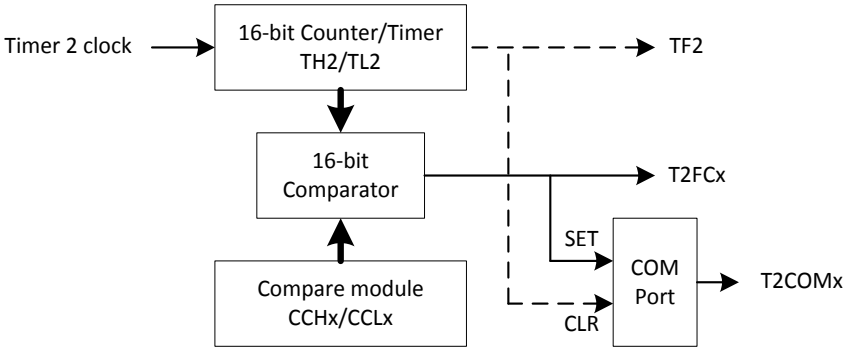
On the other hand, a falling edge of external pin T2RL (shared with P2.0) can also be chosen as a reload signal. In this situation, Timer 2 normally counts its counter from 0x0000 to 0xFFFF if T2RL pin remains stable, yet the counter value would be replaced at any time by CRCH/CRCL value as long as T2RL pin has a falling signal. Subsequently, Timer 2 continues its counting routine from CRCH/CRCL value, and external reload flag (TF2RL) would be issued if interrupt function is enabled (both ET2RL and ET2 are set). External reload interrupt vector is shared with Timer 2 interrupt vector and identify event by firmware.

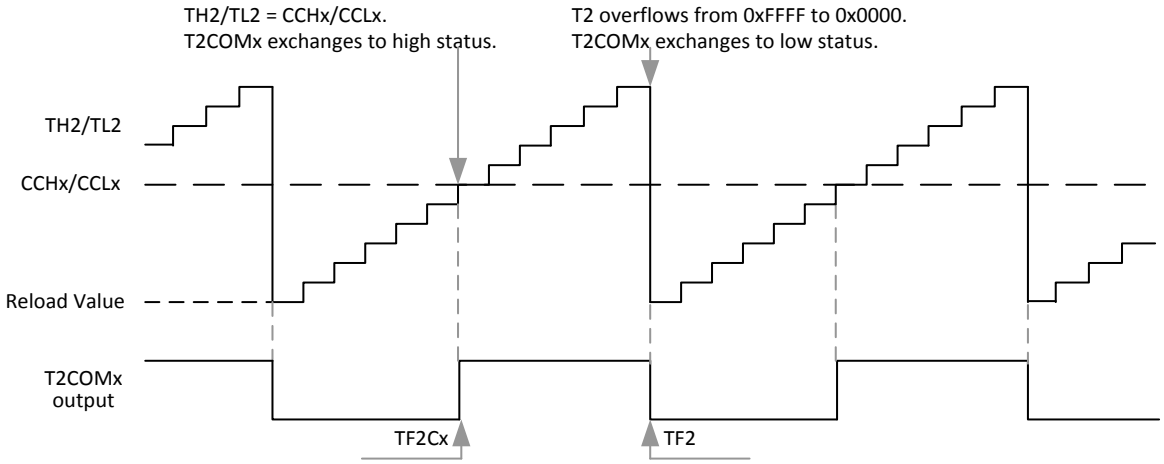


## 14.3 Comparison Output (PWM)

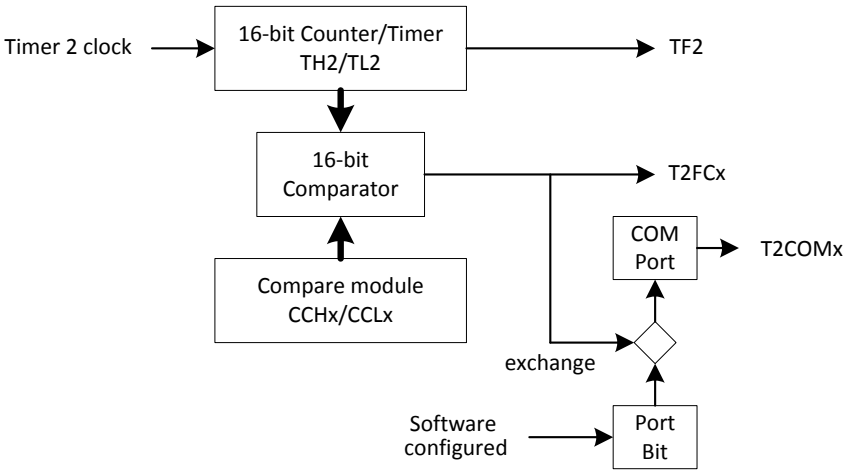
Timer 2 has up to four set of comparison output. Each set (CRC/CC1/CC2/CC3) independently compares its value to Timer 2 counter (TH2/TL2) and outputs the comparison result on T2COM0 to T2COM3 pins (shared with P00, P01, P03 and P0.4). The comparison result has two output methods: directly output and indirectly output.

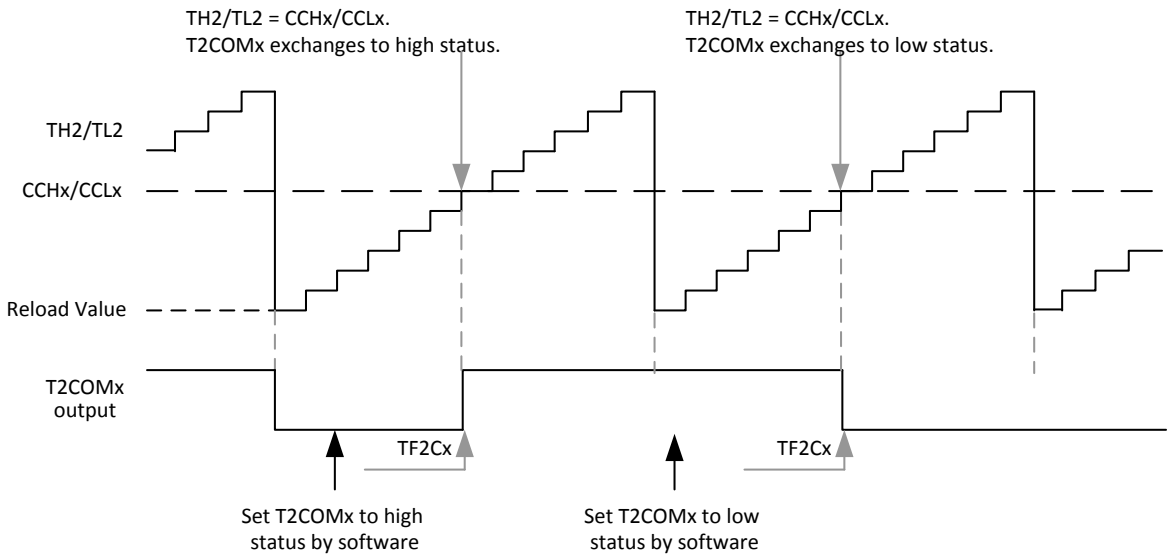
The directly method is that the mapped pin outputs low status if Timer 2 counter is lower than CRC/CC1/CC2/CC3 register, whereas it outputs high status if Timer 2 counter is equal/lager than CRC/CC1/CC2/CC3 register. Thus, the output status is changed twice at crossover points. As CRC/CC1/CC2/CC3 register is equal to Timer 2 counter, a TF2C0/TF2C1/TF2C2/TF2C3 flag is issued which can read/write by firmware. Compare interrupt function is controlled by ET2C0/ET2C1/ET2C2/ET2C3.



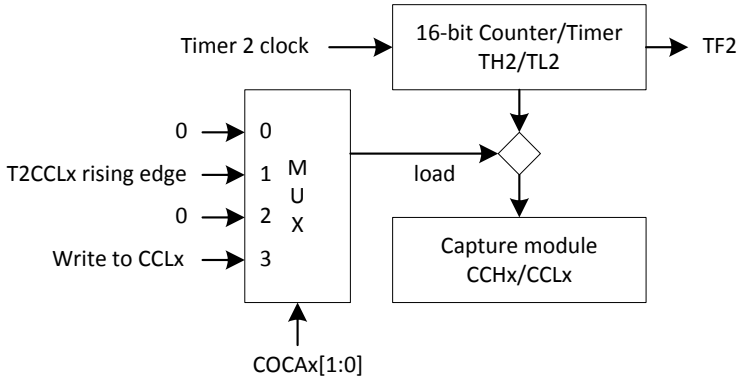

By contrast, the indirectly output method is an event which keep the mapped pin's previous output setting until Timer 2 counter overtakes CRC/CC1/CC2/CC3 register value. In this mode, the transition of the output signal can be configured by software. In other word, the P0.0 register bit would be affect T2COM0/P0.0 pin when TH2/TL2 equal to CRC registers. A Timer 2 overflow causes no output change.



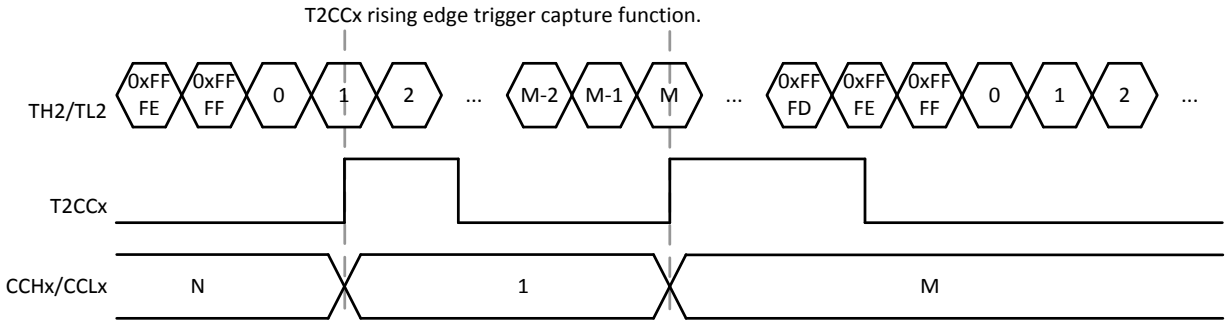




## 14.4 Capture Function

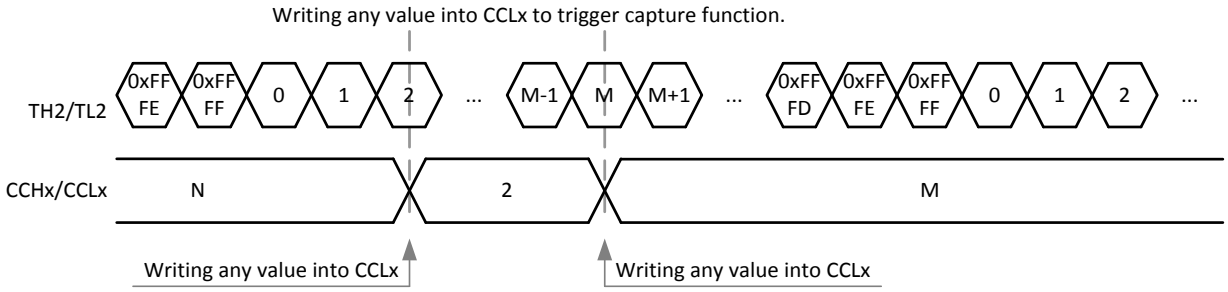
The capture function is similar to split/lap button of a stopwatch. While Timer 2 counter (TH2/TL2) routinely count up, a split event records counter value in CRC/CC1/CC2/CC3 register(s).



The split event can from hardware or software. The T2CC0 pin can trigger a hardware split event that duplicates TH2/TL2 value to CRCH/CRCL registers, whereas T2CC1, T2CC2 and T2CC3 respectively control CC1 to CC3 registers.




A software split event is triggered by writing any value into CRCL/CCL1/CCL2/CCL3 register. While perform a writing instruction to these registers, the present TH2/TL2 value would be record in the paired registers instead.




## 14.5 Timer 2 Registers

| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| T2CON | T2PS | I3FR | - | T2R1 | T2R0 | T2CM | T2I1 | T2I0 |
| CCEN | COCA31 | COCA30 | COCA21 | COCA20 | COCA11 | COCA10 | COCA01 | COCA00 |
| TH2 | TH27 | TH26 | TH25 | TH24 | TH23 | TH22 | TH21 | TH20 |
| TL2 | TL27 | TL26 | TL25 | TL24 | TL23 | TL22 | TL21 | TL20 |
| CRCH | CRCH7 | CRCH6 | CRCH5 | CRCH4 | CRCH3 | CRCH2 | CRCH1 | CRCH0 |
| CRCL | CRCL7 | CRCL6 | CRCL5 | CRCL4 | CRCL3 | CRCL2 | CRCL1 | CRCL0 |
| CCH3 | CCH37 | CCH36 | CCH35 | CCH34 | CCH33 | CCH32 | CCH31 | CCH30 |
| CCL3 | CCL37 | CCL36 | CCL35 | CCL34 | CCL33 | CCL32 | CCL31 | CCL30 |
| CCH2 | CCH27 | CCH26 | CCH25 | CCH24 | CCH23 | CCH22 | CCH21 | CCH20 |
| CCL2 | CCL27 | CCL26 | CCL25 | CCL24 | CCL23 | CCL22 | CCL21 | CCL20 |
| CCH1 | CCH17 | CCH16 | CCH15 | CCH14 | CCH13 | CCH12 | CCH11 | CCH10 |
| CCL1 | CCL17 | CCL16 | CCL15 | CCL14 | CCL13 | CCL12 | CCL11 | CCL10 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| IEN0 | EAL | - | ET2 | ES0 | ET1 | - | ET0 | EX0 |
| IEN1 | ET2RL | - | ET2C3 | ET2C2 | ET2C1 | ET2C0 | ESPI | EI2C |
| IRCON | TF2RL | TF2 | TF2C3 | TF2C2 | TF2C1 | TF2C0 | - | - |

**T2CON Register (0xC8)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | T2PS | R/W | 0 | Timer 2 pre-scalar<br>0: fcpu/12<br>1: fcpu/24 |
| 6 | I3FR | R/W | 0 | In compare mode:<br>0: The COM0 interrupt would be generated when the content of Timer2 become not equal to the CRC register.<br>1: The COM0 interrupt would be generated when the content of Timer2 become equal to the CRC register.<br>In capture mode 0:<br>0: The timer 2 content would be latched into CRC register by T2CC0 is falling edge.<br>1: The timer 2 content would be latched into CRC register by T2CC0 is rising edge. |
| 5 | Reserved | R/W | 0 | |
| 4..3 | T2R[1:0] | R/W | 00 | Specified Timer 2 reload value<br>00: Disable<br>01: Disable<br>10: Load CRCH/CRCL to TH2/TL2 by counter overflow<br>11: Load CRCH/CRCL to TH2/TL2 by T2RL pin |
| 2 | T2CM | R/W | 0 | Timer 2 comparison output<br>0: Directly output method<br>1: Indirectly output, next output status can be specified |
| 1..0 | T2I[1:0] | R/W | 00 | Timer 2 up counting control<br>00: Disable<br>01: Clock rate is defined by T2PS<br>10: Clock source is T2 pin<br>11: Clock rate is defined by T2PS with T2 pin gate control |

**CCEN Register (0xC1)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..6 | COCA3[1:0] | R/W | 00 | Comparison and capture function of CC3<br>00: Disable<br>01: Capture by T2CC3 pin rising edge<br>10: Comparison function<br>11: Capture by writing CCL3 register |
| 5..4 | COCA2[1:0] | R/W | 00 | Comparison and capture function of CC2<br>00: Disable<br>01: Capture by T2CC2 pin rising edge<br>10: Comparison function<br>11: Capture by writing CCL2 register |
| 3..2 | COCA1[1:0] | R/W | 00 | Comparison and capture function of CC1<br>00: Disable<br>01: Capture by T2CC1 pin rising edge<br>10: Comparison function<br>11: Capture by writing CCL1 register |
| 1..0 | COCA0[1:0] | R/W | 00 | Comparison and capture function of CRC<br>00: Disable<br>01: Capture by T2CC0 pin rising edge<br>10: Comparison function<br>11: Capture by writing RCCL register |

**TH2/TL2 Registers (TH2: 0xCD, TL2: 0xCC)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | TH2/TL2 | R/W | 0x00 | Timer 2 16-bit counter registers. |

**CRC Registers (CRCH: 0xCB, CRCL: 0xCA)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..6 | CRCH[15:0] | R/W | 0x00 | 16-bit compare/capture registers. |

**CCH3/CCL3 Registers (CCH3: 0xC7, CCL3: 0xC6)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..6 | CCH3/CCL3 | R/W | 0x00 | 16-bit compare/capture registers. |

**CCH2/CCL2 Registers (CCH2: 0xC5, CCL2: 0xC4)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..6 | CCH2 /CCL2 | R/W | 0x00 | 16-bit compare/capture registers. |

**CCH1/CCL1 Registers (CCH1: 0xC3, CCL1: 0xC2)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..6 | CCH1/CCL1 | R/W | 0x00 | 16-bit compare/capture registers. |

**IEN0 Register (0xA8)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | EAL | R/W | 0 | Interrupts enable. Refer to Chapter Interrupt |
| 5 | ET2 | R/W | 0 | Enable Timer 2 interrupt |
| Else | | | | Refer to other chapter(s) |

**IEN1 Register (0xB8)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | ET2RL | R/W | 0 | T2 Timer external reload interrupt control bit<br>0: Disable<br>1: Enable |
| 6 | Reserved | R | 0 | |
| 5 | ET2C3 | R/W | 0 | T2 Timer COM3 interrupt control bit<br>0: Disable<br>1: Enable |
| 4 | ET2C2 | R/W | 0 | T2 Timer COM2 interrupt control bit<br>0: Disable<br>1: Enable |
| 3 | ET2C1 | R/W | 0 | T2 Timer COM1 interrupt control bit<br>0: Disable<br>1: Enable |
| 2 | ET2C0 | R/W | 0 | T2 Timer COM0 interrupt control bit<br>0: Disable<br>1: Enable |
| Else | | | | Refer to other chapter(s) |

## 14.6 Sample Code

The following sample code demonstrates how to perform T2 compare function with interrupt.

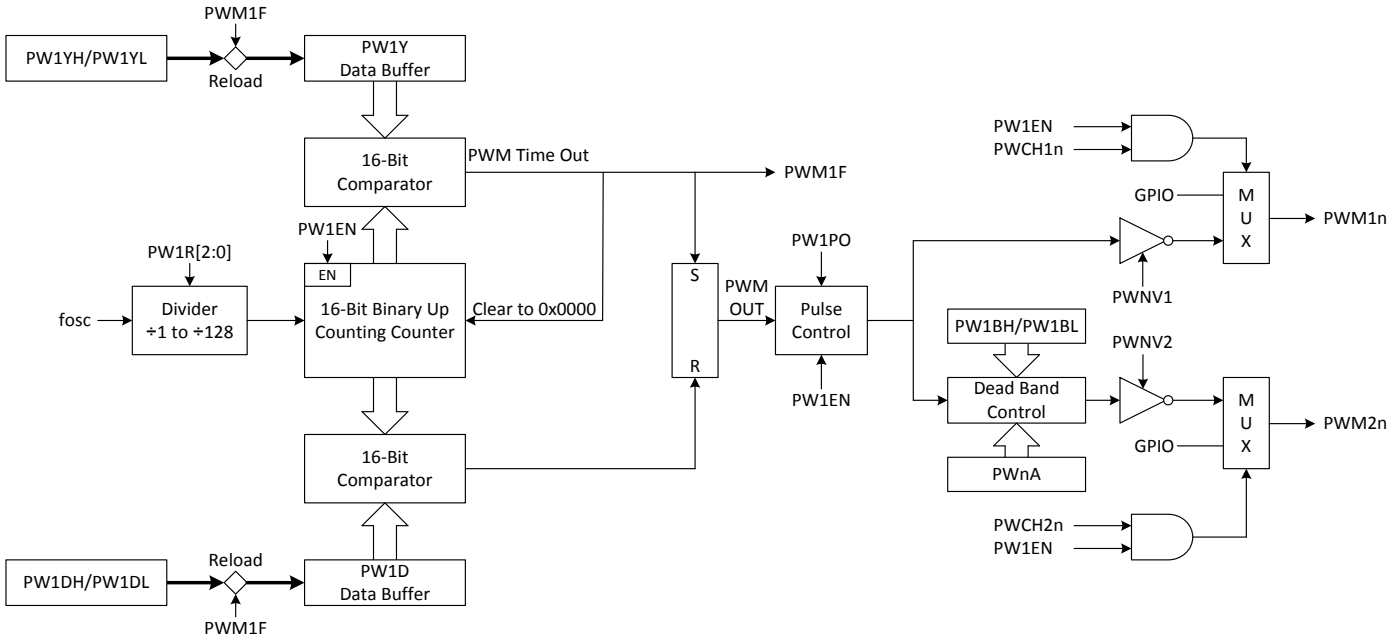
```
#define T2ClkFcpu    (1 << 0)  //T2 clock from Fcpu
#define T2ClkPin     (2 << 0)  //T2 clock from T2 pin
#define T2ClkGate    (3 << 0)  //T2 clock from Fcpu with T2 pin gating
#define T2Fcpu12     (0 << 7)  //T2 clock = Fcpu/12
#define T2Fcpu24     (1 << 7)  //T2 clock = Fcpu/24
#define T2RLMode0    (2 << 3)  //T2 reload mode0 = auto-reload
#define T2RLMode1    (3 << 3)  //T2 reload mode1 = T2RL falling edge trigger
#define ComMode0     (0 << 2)  //Compare mode = directly method
#define ComMode1     (1 << 2)  //Compare mode = indirectly output method
#define T2COM0EdNE   (0 << 6)  //T2COM0 interrupt edge = no equle CRC
#define T2COM0EdE    (1 << 6)  //T2COM0 interrupt edge = equle CRC
#define T2COM0En     (2 << 0)  //T2COM0 compare funcion enable
#define T2COM1En     (2 << 2)  //T2COM1 compare funcion enable
#define T2COM2En     (2 << 4)  //T2COM2 compare funcion enable
#define T2COM3En     (2 << 6)  //T2COM3 compare funcion enable

void InitT2(void)
{
  // T2_Initial
  TH2 = 0x00;
  TL2 = 0x00;
  CRCH = 0x80;
  CRCL = 0x00;
  CCH1 = 0xC0;
  CCL1 = 0x00;
  CCH2 = 0xE0;
  CCL2 = 0x00;
  CCH3 = 0xF0;
  CCL3 = 0x00;

  // T2 clock from Fcpu/24 with T2 pin gating
  // Reload mode1 = T2RL falling edge trigger
  // Compare mode = directly method
  // T2COM0 interrupt trigger = equle CRC
  T2CON |= T2ClkGate | T2Fcpu24 | T2RLMode1 | ComMode0 | T2COM0EdE;

  // Compare function T2COM0/1/2/3 enable
  CCEN |= T2COM0En | T2COM1En | T2COM2En | T2COM3En;

  // P07(T2)/P20(T2RL) is input mode with pull-high resister
  P0M &= 0x7F;
  P2M &= 0xFE;
  P0UR &= 0x80;
  P2UR &= 0x01;

  // Enable T2RL/T2COM0/1/2/3 interrupt
  IEN1 |= 0xBC;

  // Enable total/Timer2 interrupt
  IEN0 |= 0xA0;

  P0 = 0x00;
```

```
53 }
54
55 void T2Interrupt(void) interrupt ISRTimer2  //0x2B
56 { //TF2/TF2RL clear by software
57   if ((IRCON & 0x40) == 0x40) {
58     IRCON  &= 0xBF;   //Clear TF2
59     P00 = ~P00;
60   }
61   if ((IRCON & 0x80) == 0x80) {
62     IRCON  &= 0x7F;   //Clear TF2RL
63     P01 = ~P01;
64   }
65 }
66
67 void T2COM0Interrupt(void) interrupt ISRCom1  //0x53
68 { //TF2C0 clear by hardware
69   P02 = ~P02;
70 }
71
72 void T2COM1Interrupt(void) interrupt ISRCom2  //0x5B
73 { //TF2C1 clear by hardware
74   P03 = ~P03;
75 }
76
77 void T2COM2Interrupt(void) interrupt ISRCom3  //0x63
78 { //TF2C2 clear by hardware
79   P04 = ~P04;
80 }
81
82 void T2COM3Interrupt(void) interrupt ISRCom4  //0x6B
83 { //TF2C3 clear by hardware
84   P05 = ~P05;
85 }
```

# 15 PWM

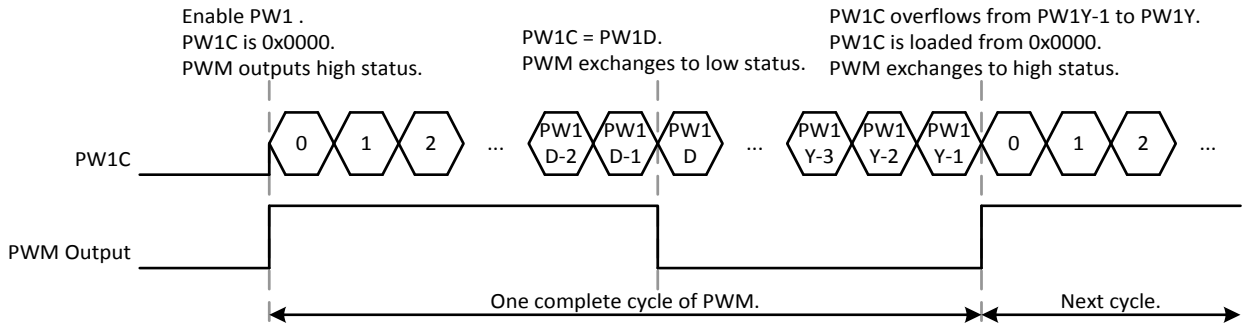
The PW1 timer includes a 16-bit binary up 4-channel PWM, and one pulse PWM functions. By the counter reaches the up-boundary value (PW1Y), it clears its counter and triggers an interrupt signal. PWM's duty cycle is controlled by PW1D register.

The PWM also support one pulse output signal which can disables itself by the end of first PWM cycle. Thus, only one pulse would be generated in this condition. The PWM has four programmable channels shared with GPIO pins and controlled by PW1CH register. The output operation must be through enabled each bit/channel of PW1CH register. The enabled PWM channel exchanges from GPIO to PWM output. When the PW1CH register disables, the PWM channel returns to last status of GPIO mode. The PWM build in IDLE Mode wake-up function if interrupt enable. When PWM timer overflow occurs (counts from PW1Y-1 to PW1Y), PWM1F would be issued immediately which can read/write by firmware. PWM clock source is fosc, and divided by 1 to 128 times which is controlled by PW1R[2:0] bits. PW1 interrupt function is controlled by EPWM1



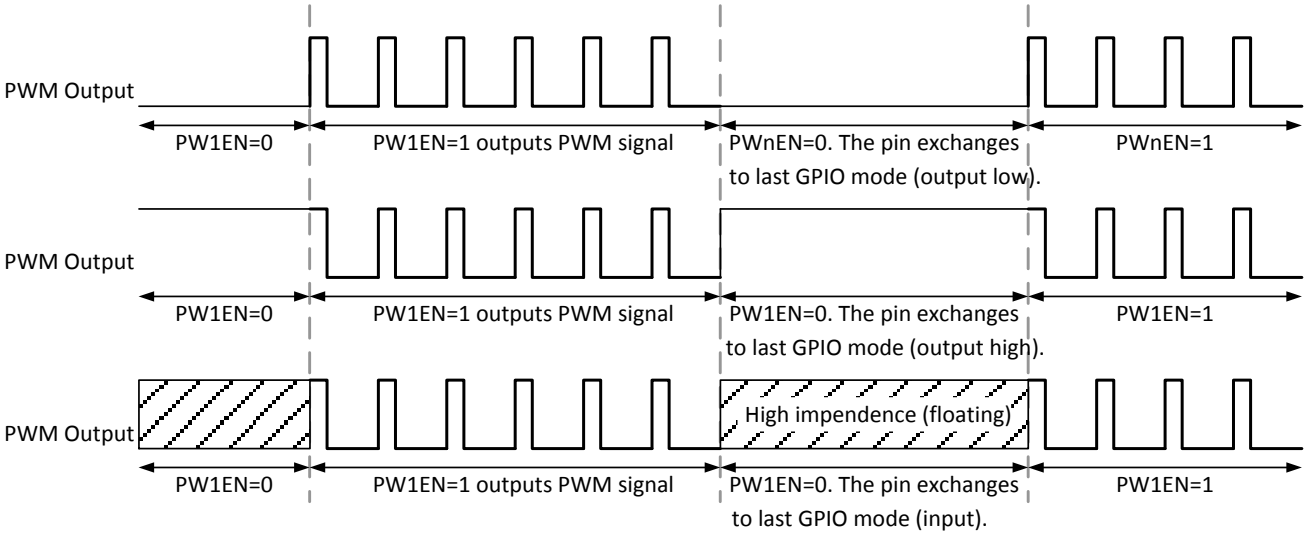
## 15.1 General PWM

PW1 timer builds in PWM function controlled by PW1EN and PW1CH register. PWM10, PWM11, PWM20, PWM21 are output pins (shared with P0.5, P0.6, P1.4, and P1.5). The PWM output pins are shared with GPIO pin controlled by PW1CH register. When output PWM function, we must be set PW1EN =1. When PWM output signal synchronize finishes, the PWM channel exchanges from GPIO to PWM output. When PW1EN = 0, the PWM channel returns to GPIO mode and last status. PWM signal is generated from the result of PW1Y and PW1D comparison combination. When PW1C counts from 0x0000, the PWM outputs high status which is the PWM initial status. PW1C is loaded new data from PW1Y register to decide PWM cycle and resolution. PW1C keeps counting, and the system compares PW1C and PW1D. When PW1C=PW1D, the PWM output status exchanges to low PW1C keeps counting. When PW1 timer overflow occurs (PW1Y-1 to 0x0000), and one cycle of PWM signal finishes. PW1C is reloaded from 0x0000 automatically, and PWM output status exchanges to high for next cycle. PW1D decides the high duty duration, and PW1Y decides the resolution and cycle of PWM. PW1D can't be larger than PW1Y, or the PWM signal is error.



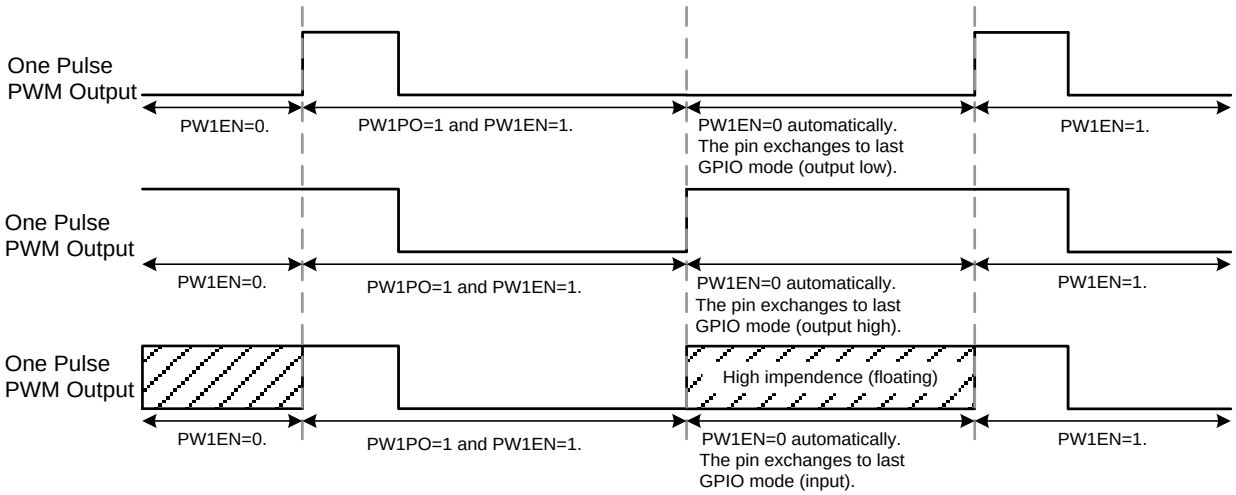

PWM Period = PW1Y

PWM duty = (PW1D): (PW1Y-PW1D)



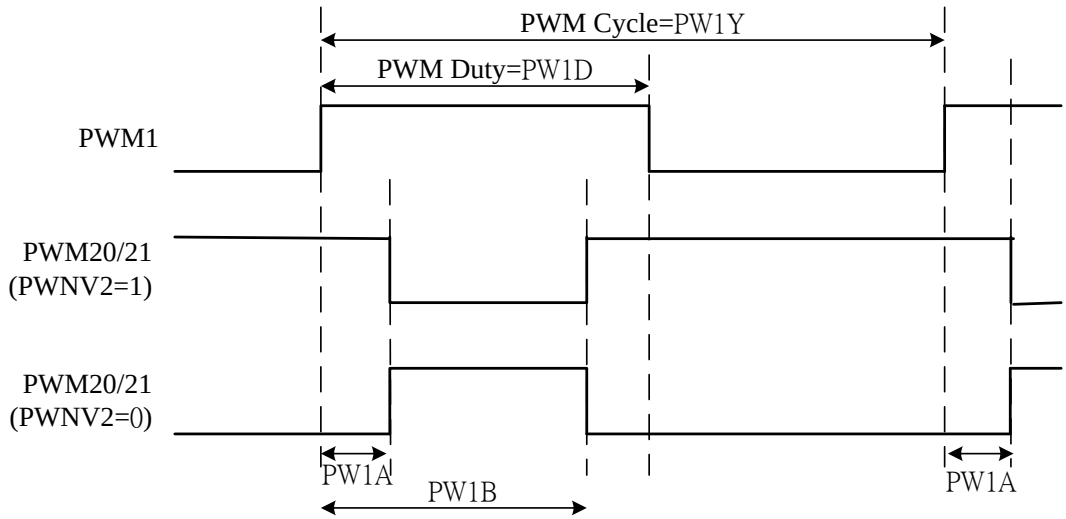
## 15.2 One Pulse PWM

When PW1PO = 0, PW1 is PWM function mode. When PW1PO = 1 and PW1EN = 1, PW1 will output one pulse PWM function and the PWM1F is issued as PW1 counter overflow. PW1EN bit is cleared automatically and pulse output pin returns to idle status. To output next pulse is to set PW1EN bit by program again. One pulse PWM channels selected by PW1CH register. When output one pulse PWM function, we must be set PW1PO = 1 and PW1EN=1. When one pulse PWM output signal synchronize finishes, the PWM channel exchanges from GPIO to PWM output. When one pulse PWM output finishes, PW1EN = 0, the PWM channel returns to GPIO mode and last status.




## 15.3 Inverse and Dead Band

The PWM builds in inverse output function. The PWM has one inverse PWM signal as PWNVn =1. When PWNVn= 1, the PW1 outputs the inverse PWM signal of PWM1. When PWNVn = 0, the PW1 outputs the non-inverse PWM signal of PWM1. The inverse PWM output waveform is below diagram.



The PWM dead band occurs in PWM high pulse width, and the dead band period is programmable from PW1A and PW1D-PW1B registers. The dead band period is symmetrical at left-right terminal of PWM pulse width or not. If the bead band period is longer than PWM duty, the PWM is no output.

## 15.4 PWM Registers

| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PW1M | PW1EN | PW1R2 | PW1R1 | PW1R0 | PWNV2 | PWNV1 | - | PW1PO |
| PW1CH | - | - | PWCH21 | PWCH20 | - | - | PWCH11 | PWCH10 |
| PW1YH | PW1Y15 | PW1Y14 | PW1Y13 | PW1Y12 | PW1Y11 | PW1Y10 | PW1Y9 | PW1Y8 |
| PW1YL | PW1Y7 | PW1Y6 | PW1Y5 | PW1Y4 | PW1Y3 | PW1Y2 | PW1Y1 | PW10 |
| PW1BH | PW1B15 | PW1B14 | PW1B13 | PW1B12 | PW1B11 | PW1B10 | PW1B9 | PW1B8 |
| PW1BL | PW1B7 | PW1B6 | PW1B5 | PW1B4 | PW1B3 | PW1B2 | PW1B1 | PW1B0 |
| PW1DH | PW1D15 | PW1D14 | PW1D13 | PW1D12 | PW1D11 | PW1D10 | PW1D9 | PW1D8 |
| PW1DL | PW1D7 | PW1D6 | PW1D5 | PW1D4 | PW1D3 | PW1D2 | PW1D1 | PW1D0 |
| PW1A | PW1A7 | PW1A6 | PW1A5 | PW1A4 | PW1A3 | PW1A2 | PW1A1 | PW1A0 |

**PW1M Registers (PW1M: 0xAB)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | PW1EN | R/W | 0 | PW1 function<br>0: Disable<br>1: Enable* |
| 6..4 | PW1R[2:0] | R/W | 000 | PWM timer clock source<br>000: fosc / 128<br>001: fosc / 64<br>010 fosc / 32<br>011: fosc / 16<br>100: fosc / 8<br>101: fosc / 4<br>110: fosc / 2<br>111: fosc / 1 |
| 3 | PWNV2 | R/W | 0 | PWM20/21 pin output control<br>0: Non-inverse<br>1: Inverse |
| 2 | PWNV1 | R/W | 0 | PWM10/11 pin output control<br>0: Non-inverse<br>1: Inverse |

| | | | | |
|---|---|---|---|---|
| 1 | Reserved | R/W | 0 | |
| 0 | PW1PO | R/W | 0 | One pulse function<br>0: Disable<br>1: Enable |

* When the period is setting 0x0000, after PWM is set enable bit, the PWM will stop and the period can't update.

**PW1CH Register (0xBE)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..6 | Reserved | R/W | 0 | |
| 5<br>4 | PWCH21<br>PWCH20 | R/W | 0 | PWM1 shared-pin control<br>0: GPIO<br>1: PWM output (shared with P0.6/ P1.5) |
| 3..2 | Reserved | R/W | 0 | |
| 1<br>0 | PWCH11<br>PWCH10 | R/W | 0 | PWM1 shared-pin control<br>0: GPIO<br>1: PWM output (shared with P0.5/ P1.4) |

**PW1YH/PW1YL Registers (PW1YH: 0xAD, PW1YL: 0xAC)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | PW1YH/L | R/W | 0x00 | 16-bit PWM1 period control*. |

* The period configuration must be setup completely before starting PWM function.

**PW1DH/PW1DL Registers (PW1DH: 0xBC, PW1DL: 0xBB)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | PW1DH/L | R/W | 0x00 | 16-bit PWM1 duty control |

**PW1BH/PW1BL Registers (PW1BH: 0xAF, PW1BL: 0xAE)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | PW1BH/L | R/W | 0x00 | 16-bit PWM1 dead band control |

**PW1A Registers (PW1A: 0xBD)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | PW1A | R/W | 0x00 | 8-bit PWM1 dead band control |

**IEN0 Register (0xA8)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | EAL | R/W | 0 | Interrupts enable. Refer to Chapter Interrupt |
| Else | | | | Refer to other chapter(s) |

**IEN4 Register (0XD1)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | EPWM1 | R/W | 0 | PWM1 interrupt control bit.<br>0 = Disable PWM1 interrupt function.<br>1 = Enable PWM1 interrupt function. |
| 3 | PWM1F | R/W | 0 | PWM1 interrupt request flag.<br>0: None PWM1 interrupt request<br>1: PWM1 interrupt request. |
| Else | Reserved | R | 0 | |

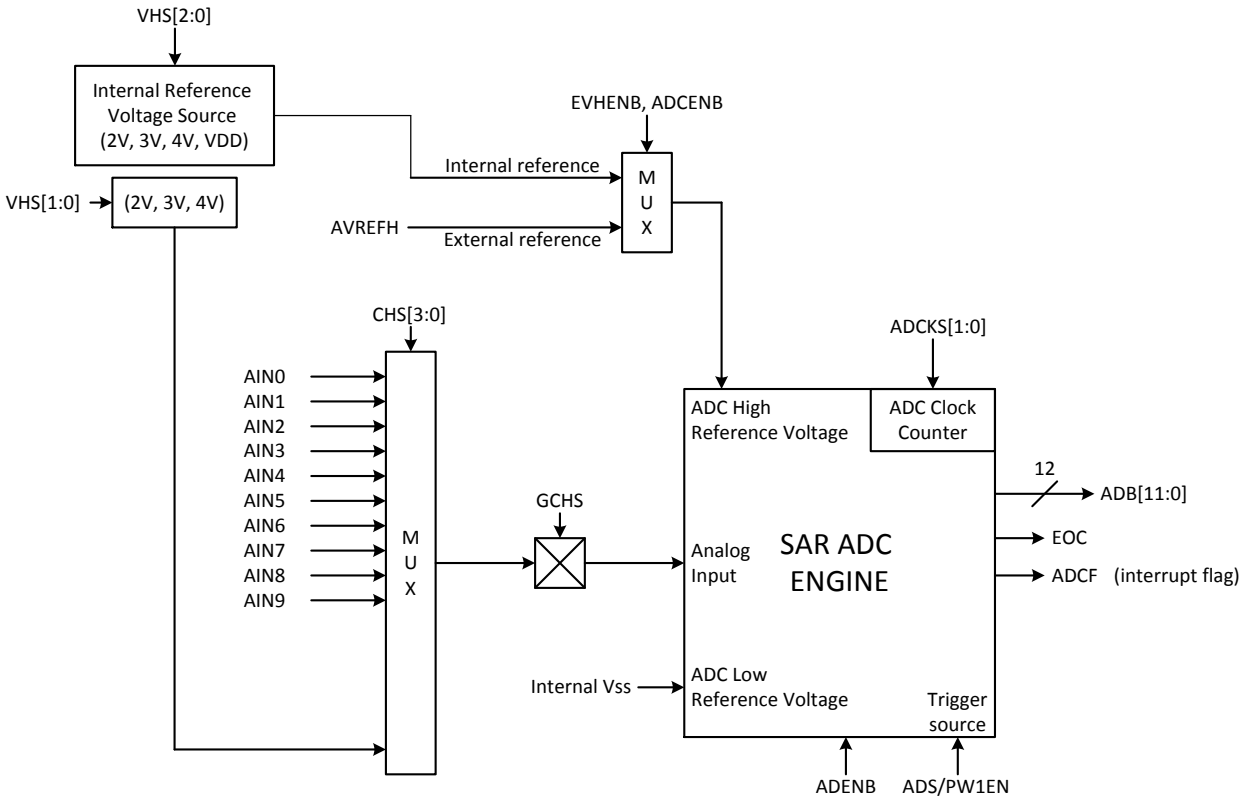
## 15.5 Sample Code

The following sample code demonstrates how to perform PW1 with interrupt.

```
#define PW1Inv1      (1 << 2)  //PWM10/11 output inverse
#define PW1Inv2      (1 << 3)  //PWM20/21 output inverse
#define PW1OnePu     (1 << 0)  //Enable PW1 pulse output function
#define PWM10En      (1 << 0)  //Enable PWM10 output function
#define PWM11En      (2 << 0)  //Enable PWM11 output function
#define PWM20En      (1 << 4)  //Enable PWM20 output function
#define PWM21En      (2 << 4)  //Enable PWM21 output function
#define PW1En       (1 << 7)  //Enable PWM1 function

void InitPWM(void)
{
  // PWM1_Initial
  PW1YH = 0x80;
  PW1YL = 0x00;
  PW1DH = 0x40;
  PW1DL = 0x00;
  PW1BH = 0x60;
  PW1BL = 0x00;
  PW1A = 0x80;

  // PW10/11/20/21 channel enable
  PW1CH = PWM10En | PWM11En | PWM20En | PWM21En ;

  // PWM1 enable, P10/11 output inverse, clock = Fosc/32
  PW1M = PW1En | PW1Inv1 | 0x20;

  // Enable PWM1 interrupt & clear PWM1F
  IEN4 = 0x80;

  // Enable total interrupt
  IEN0 |= 0x80;

  P0 = 0x00;
  P0M |= 0x01;
}

void PW1Interrupt(void) interrupt ISRPwm1  //0x83
{ //PWM1F clear by software
  if ((IEN4 & 0x08) == 0x08) {
    IEN4  &= 0xF7;   //Clear PWM1F
    P00 = ~P00;
  }
}
```

## 16 ADC

The analog to digital converter (ADC) is SAR structure with 10-input sources and up to 4096-step resolution to transfer analog signal into 12-bits digital buffers. The ADC builds in 10-channel input source to measure 10 different analog signal sources. The ADC resolution is 12-bit. The ADC has four clock rates to decide ADC converting rate. The ADC reference high voltage includes 5 sources. Four internal power source including VDD, 4V, 3V and 2V. The other one is external reference voltage input pin from AVREFH pin. The ADC builds in P1CON/P2CON registers to set pure analog input pin. After setup ADENB and ADS bits, the ADC starts to convert analog signal to digital data. Besides ADS bit can start to convert analog signal, PW1EN also have convert analog signal ADC function. ADC can work in idle mode. After ADC operating, the system would be waked up from green mode to normal mode if interrupt enable.



## 16.1 Configurations of Operation

These configurations must be setup completely before starting ADC converting. ADC is configured using the following steps:

1. Choose and enable the start of conversion ADC input channel. (By CHS[3:0] bits and GCHS bit)
2. The GPIO mode of ADC input channel must be set as input mode. (By PnM register)
3. The internal pull-up resistor of ADC input channel must be disabled. (By PnUR register)
4. The configuration control bit of ADC input channel must be set. (By PnCON register)
5. Choose ADC high reference voltage. (By VREFH register)
6. Choose ADC Clock Rate. (By ADCKS[1:0] bits)
7. After setup ADENB bits, the ADC ready to convert analog signal to digital data.

When ADC IP is enabled by ADENB bit, it is necessary to make an ADC start-up by program. Writing a 1 to the ADS bit of register ADM. After setup ADENB and ADS bits, the ADC starts to convert analog signal to digital data. The ADS bit is reset to logic 0 when the conversion is complete. When the conversion is complete, the ADC circuit will set EOC and ADCF bits to “1” and the digital data outputs in ADB and ADR registers. If ADC interrupt function is enabled (EADC = 1), the ADC interrupt request occurs and executes interrupt service routine when ADCF is “1” after ADC converting. Clear ADCF by hardware automatically in interrupt procedure. Note that when ADPWS bit is “1”, if PWM enable trigger be used as the conversion source, the ADC will continuous conversions until PWM is disabled.

## 16.2 ADC input channel

The ADC builds in 10-channel input source (AIN0 – AIN9) to measure 10 different analog signal sources controlled by CHS[3:0] and GCHS bits. The AIN10 is internal 2V or 3V or 4V input channel. There is no any input pin from outside. In this time ADC reference voltage must be internal VDD and External voltage, not internal 2V or 3V or 4V. AIN10 can be a good battery detector for battery system. To select appropriate internal AVREFH level and compare value, a high performance and cheaper low battery detector is built in the system.

ADC input channel

| CHS[3:0] | Channel | Pin name | Remark |
|---|---|---|---|
| 0000 | AIN0 | P1.0 | - |
| 0001 | AIN1 | P1.1 | - |
| 0010 | AIN2 | P1.2 | - |
| 0011 | AIN3 | P1.3 | - |
| 0100 | AIN4 | P1.4 | - |
| 0101 | AIN5 | P1.5 | - |
| 0110 | AIN6 | P1.6 | - |
| 0111 | AIN7 | P1.7 | - |
| 1000 | AIN8 | P2.1 | - |
| 1001 | AIN9 | P2.0 | - |
| 1010 | AIN10 | Internal 2V or 3V or 4V | Battery detector channel |
| 1011 – 1111 | - | - | Reserved |

### 16.2.1 Pin Configuration

ADC input channels are shared with Port1 and Port2. ADC channel selection is through CHS[3:0] bit. Only one pin of Port1 and Port2 can be configured as ADC input in the same time. The pins of Port1 and Port2 configured as ADC input channel must be set input mode, disable internal pull-up and enable P1CON/P2CON first by program. After selecting ADC input channel through CHS[3:0], set GCHS bit as “1” to enable ADC channel function.

ADC input pins are shared with digital I/O pins. Connect an analog signal to COMS digital input pin, especially, the analog signal level is about 1/2 VDD will cause extra current leakage. In the power down mode, the above leakage current will be a big problem. Unfortunately, if users connect more than one analog input signal to Port1 or Port2 will encounter above current leakage situation. Write “1” into PnCON register will configure related pin as pure analog input pin to avoid current leakage.

Note that When ADC pin is general I/O mode, the bit of P1CON and P2CON must be set to “0”, or the digital I/O signal would be isolated.

## 16.3 Reference Voltage

The ADC builds in five high reference voltage source controlled through VREFH register. There are one external voltage source and four internal voltage source (VDD, 4V, 3V, 2V). When EVHENB bit is “1”, ADC reference voltage is external voltage source from AVREFH/P1.0. In the condition, P1.0 GPIO mode must be set as input mode and disable internal pull-up resistor.

If EVHENB bit is “0”, ADC reference high voltage is from internal voltage source selected by VHS[1:0] bits. If VHS[1:0] is “11”, ADC reference high voltage is VDD. If VHS[1:0] is “10”, ADC reference high voltage is 4V. If VHS[1:0] is “01”, ADC reference high voltage is 3V. If VHS[1:0] is “00”, ADC reference high voltage is 2V. The limitation of internal high reference voltage application is VDD can’t below each of internal high voltage level, or the level is equal to VDD. If AIN10 channel is selected as internal 2V or 3V or 4V input channel. There is no any input pin from outside. In this time ADC reference high voltage must be internal VDD or External voltage, not internal 2V/3V/4V.

### 16.3.1 Signal Format

ADC sampling voltage range is limited by high/low reference voltage. The ADC low reference voltage is Vss and not changeable. The ADC high reference voltage includes internal VDD/4V/3V/2V and external reference voltage source from P1.0/AVREFH pin controlled by EVHENB bit. ADC reference voltage range limitation is “(ADC high reference voltage - low reference voltage) ≥ 2V”. ADC low reference voltage is Vss = 0V. So ADC high reference voltage range is 2V to VDD. The range is ADC external high reference voltage range.

- ADC Internal Low Reference Voltage = 0V.
- ADC Internal High Reference Voltage = VDD/4V/3V/2V. (EVHENB=0)
- ADC External High Reference Voltage = 2V to VDD. (EVHENB=1)

ADC sampled input signal voltage must be from ADC low reference voltage to ADC high reference. If the ADC input signal voltage is over the range, the ADC converting result is error (full scale or zero).

- ADC Low Reference Voltage ≤ ADC Sampled Input Voltage ≤ ADC High Reference Voltage

## 16.4 Converting Time

The ADC converting time is from ADS=1 (Start to ADC convert) to EOC=1 (End of ADC convert). The converting time duration is depend on ADC clock rate. 12-bit ADC’s converting time is
1/ (ADC clock /4)*16 sec. ADC clock source is fosc and includes fosc/1, fosc/2, fosc/8 and fosc/16 controlled by ADCKS[1:0] bits.

The ADC converting time affects ADC performance. If input high rate analog signal, it is necessary to select a high ADC converting rate. If the ADC converting time is slower than analog signal variation rate, the ADC result would be error. So to select a correct ADC clock rate to decide a right ADC converting rate is very important.

$$12 \text{ bits ADC conversion time} = \frac{16}{\text{ADC clock rate}/4}$$

ADC converting time

| ADCKS[1:0] | ADC clock rate | fosc = 32MHz | |
|---|---|---|---|
| | | Converting time | Converting rate |
| 00 | fosc/16 | 1/(32MHz/16/4)*16 = 32us | 31.25kHz |
| 01 | fosc/8 | 1/(32MHz/8/4)*16 = 16us | 62.5kHz |
| 10 | fosc | 1/(32MHz/4)*16 = 2us | 500kHz |
| 11 | fosc/2 | 1/(32MHz/2/4)*16 = 4us | 250kHz |

## 16.5 Data Buffer

ADC data buffer is 12-bit length to store ADC converter result. The high byte is ADB register, and the low-nibble is ADR[3:0] bits. The ADB register is only 8-bit register including bit 4 – bit 11 ADC data. To combine ADB register and the low-nibble of ADR will get full 12-bit ADC data buffer. The ADC data buffer is a read-only register and the initial status is unknown after system reset.

Table 16-1 The AIN input voltage vs. ADB output data

| AIN n | ADB11 | ADB10 | ADB9 | ADB8 | ADB7 | ADB6 | ADB5 | ADB4 | ADB3 | ADB2 | ADB1 | ADB0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0/4096*VREFH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1/4096*VREFH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| . | . | . | . | . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . | . | . | . | . |
| 4094/4096*VREFH | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 4095/4096*VREFH | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## 16.6 ADC Registers

| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| ADM | ADENB | ADS | EOC | - | CHS3 | CHS2 | CHS1 | CHS0 |
| ADB | ADB11 | ADB10 | ADB9 | ADB8 | ADB7 | ADB6 | ADB5 | ADB4 |
| ADR | - | GCHS | ADCKS1 | ADCKS0 | ADB3 | ADB2 | ADB1 | ADB0 |
| VREFH | EVHENB | - | - | ADPWS | - | VHS2 | VHS1 | VHS0 |
| P1CON | P1CON7 | P1CON6 | P1CON5 | P1CON4 | P1CON3 | P1CON2 | P1CON1 | P1CON0 |
| P2CON | - | - | - | - | - | - | P2CON1 | P2CON0 |
| IEN2 | - | - | - | - | - | - | EADC | - |
| IRCON2 | - | - | - | | - | - | - | ADCF |

**ADM Register (0xD2)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | ADENB | R/W | 0 | ADC control bit. In stop mode, disable ADC to reduce power consumption.<br>0: Disable<br>1: Enable |
| 6 | ADS | R/W | 0 | ADC conversion control<br>Write 1: Start ADC conversion (automatically cleared by the end of conversion) |
| 5 | EOC | R/W | 0 | ADC status bit.<br>0: ADC progressing<br>1: End of conversion (automatically set by hardware; manually cleared by firmware) |
| 4 | Reserved | R | 0 | |
| 3..0 | CHS[3:0] | R/W | 0x00 | ADC input channel select bit.<br>0000: AIN0, 0001: AIN1,<br>0010: AIN2, 0011: AIN3,<br>0100: AIN4, 0101: AIN5,<br>0110: AIN6, 0111: AIN7,<br>1000: AIN8, 1001: AIN9,<br>1010: AIN10*(1), others: Reserved. |

*(1) The AIN10 is internal 2V or 3V or 4V input channel. There is no any input pin from outside. In this time ADC reference voltage must be internal VDD and External voltage, not internal 2V or 3V or 4V.

**ADB Register (0xD3)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | ADB[11:4] | R | - | ADC Result Bit [11:4]* in 12-bit ADC resolution mode. |

* ADC data buffer is 12-bit length to store ADC converter result. The high byte is ADB register, and the low-nibble is ADR[3:0] bits.

**ADR Register (0xD4)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 6 | GCHS | R/W | 0 | ADC global channel select bit.<br>0: Disable AIN channel.<br>1: Enable AIN channel. |
| 5..4 | ADCKS[1:0] | R/W | 00 | ADC's clock source select bit.<br>00 = fosc/16, 01 = fosc/8, 10 = fosc/1, 11 = fosc/2 |
| 3..0 | ADB[3:0] | R | - | ADC Result Bit [3:0]* in 12-bit ADC resolution mode. |

* ADC data buffer is 12-bit length to store ADC converter result. The high byte is ADB register, and the low-nibble is ADR[3:0] bits.

**VREFH Register (0xD5)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | EVHENB | R/W | 0 | ADC internal reference high voltage control bit.<br>0: Enable ADC internal VREFH function. AVREFH/P1.0 pin is GPIO.<br>1: Disable ADC internal VREFH function. AVREFH/P1.0 pin is external AVREFH*(1) input pin. |
| 4 | ADPWS | R/W | 0 | PWM trigger ADC start control bit.<br>0: Disable PWM trigger ADC start.<br>1: Enable PWM trigger ADC start. |
| 2 | VHS[2] | R/W | 0 | ADC internal reference high voltage select bit for AIN10.<br>0: ADC internal VREFH function is depend on VHS[1:0]*(2).<br>1: ADC internal VREFH function is internal VDD. |
| 1..0 | VHS[1:0] | R/W | 00 | ADC internal reference high voltage selects bits.<br>00: 2.0V<br>01: 3.0V<br>10: 4.0V<br>11: VDD |

*(1) The AVREFH level must be between the VDD and 2.0V.

*(2) If AIN10 channel is selected as internal 2V or 3V or 4V input channel. There is no any input pin from outside. In this time ADC reference high voltage must be internal VDD or External voltage, not internal 2V/3V/4V.

**P1CON Register (0xD6)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | P1CON[7:0] | R/W | 0x00 | P1 configuration control bit*.<br>0: P1 can be analog input pin (ADC input pin) or digital GPIO pin.<br>1: P1 is pure analog input pin and can't be a digital GPIO pin. |

* P1CON [7:0] will configure related Port1 pin as pure analog input pin to avoid current leakage.

**P2CON Register (0x9E)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 1..0 | P2CON[1:0] | R/W | 0x0 | P2 configuration control bit*.<br>0: P2 can be analog input pin (ADC input pin) or digital GPIO pin.<br>1: P2 is pure analog input pin and can't be a digital GPIO pin. |

* P2CON [1:0] will configure related Port2 pin as pure analog input pin to avoid current leakage.

**IEN2 Register (0x9A)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 1 | EADC | R/W | 0 | ADC interrupt control bit.<br>0: Disable ADC interrupt function.<br>1: Enable ADC interrupt function. |
| Else | | | | Refer to other chapter(s) |

**IRCON2 Register (0xBF)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 0 | ADCF | R/W | 0 | ADC interrupt request flag.<br>0 = None ADC interrupt request.<br>1 = ADC interrupt request. |
| Else | | | | Refer to other chapter(s) |

## 16.7 Sample Code

The following sample code demonstrates how to perform ADC to convert AIN5 with interrupt.

```
#define ADCAIN12_VDD   (3 << 0)  //AIN12 = VDD
#define ADCAIN12_4V    (2 << 0)  //AIN12 = 4.0V
#define ADCAIN12_3V    (1 << 0)  //AIN12 = 3.0V
#define ADCAIN12_2V    (0 << 0)  //AIN12 = 2.0V
#define ADCInRefVDD    (1 << 2)  //internal reference from VDD
#define ADCExHighRef   (1 << 7)  //high reference from AVREFH/P2.0
#define ADCSpeedDiv16  (0 << 4)  //ADC clock = fosc/16
#define ADCSpeedDiv8   (1 << 4)  //ADC clock = fosc/8
#define ADCSpeedDiv1   (2 << 4)  //ADC clock = fosc/1
#define ADCSpeedDiv2   (3 << 4)  //ADC clock = fosc/2
#define ADCChannelEn   (1 << 6)  //enable ADC channel
#define SelAIN5        (5 << 0)  //select ADC channel 5
#define ADCStart       (1 << 6)  //start ADC conversion
#define ADCEn          (1 << 7)  //enable ADC
#define EADC           (1 << 1)  //enable ADC interrupt
#define ClearEOC       0xDF;

unsigned int  ADCBuffer;    // data buffer

void ADCInit(void)
{
  P1  = 0x00;
  P1M = 0x80;

  // set AIN5 pin's mode at pure analog pin
  P1CON |= 0x20;     //AIN5/P15
  P1M   &= 0xDF;    //input mode
  P1UR  &= 0xDF;    //disable pull-high

  // configure ADC channel and enable ADC.
  ADM = ADCEn | SelAIN5;

  // enable channel and select conversion speed
  ADR = ADCChannelEn | ADCSpeedDiv1;

  // configure reference voltage
  VREFH = ADCInRefVDD;

  // enable gobal interrupt
  IEN0  |= 0x80;    //enable global interrupt

  // enable ADC interrupt
  IEN2  |= EADC;

  // start ADC conversion
  ADM   |= ADCStart;
}

void ADCInterrupt(void) interrupt ISRAdc
{
  if ((IRCON2 & 0x01) == 0x01) {
    P17 = ~P17;
    IRCON2  &= 0xFE;   //Clear ADCF
```

```
    ADCBuffer = (ADB << 4) + (ADR & 0x0F);
    ADM &= ClearEOC;
    ADM |= ADCStart;
  }
}
```

# 17 UART

The UART provides a flexible full-duplex synchronous/asynchronous receiver/transmitter. The serial interface provides an up to 0.25MHz flexible full-duplex transmission. It can operate in four modes (one synchronous and three asynchronous). Mode0 is a shift register mode and operates as synchronous transmitter/receiver. In Mode1-Mode3 the UART operates as asynchronous transmitter/receiver with 8-bit or 9-bit data. The transfer format has start bit, 8-bit/ 9-bit data and stop bit. Transmission is started by writing to the S0BUF register. After reception, input data are available after completion of the reception in the S0BUF register. TB80/RB80 bit can be used as the 9th bit for transmission and reception in 9-bit UART mode. Programmable baud rate supports different speed peripheral devices.

The UART features include the following:

- Full-duplex, 2-wire synchronous/asynchronous data transfer.
- Programmable baud rate.
- 8-bit shift register: operates as synchronous transmitter/receiver
- 8-bit / 9-bit UART: operates as asynchronous transmitter/receiver with 8 or 9-bit data bits and programmable baud rate.

## 17.1 UART Operation

The UART UTX and URX pins are shared with GPIO. In synchronous mode, the UTX/URX shared pins must set output high by software. In asynchronous mode (8-bit/9-bit UART), the UTX shared pins must set output high and URX set input high by software. Thus, URX/UTX pins will transfers to UART purpose. When UART disables, the UART pins returns to GPIO last status.

The UTX/URX pins also support open-drain structure. The open-drain option is controlled by PnOC bit. When PnOC=0, disable UTX/URX open-drain structure. When PnOC=1, enable UTX/URX open-drain structure. If enable open-drain structure, UTX/URX pin must set high level (IO mode control will be ignored) and need external pull-up resistor.

The UART supports interrupt function. ES0 is UART0 transfer interrupt function control bit. UART transmitter and receiver interrupt function is controlled by ES0. When ES0 = 0, disable transmitter/receiver interrupt function. When ES0 = 1, enable UART transmitter/ receiver interrupt function. UART transmitter and receiver interrupt function are share interrupt vector 0x0023. When UART interrupt function enable, the program counter points to interrupt vector to do UART interrupt service routine after UART operating. TI0/RI0 is UART0 interrupt request flag, and also to be the UART operating status indicator when interrupt is disabled. TI0 and RI0 must clear by software.

UART provides four operating mode (one synchronous and three asynchronous) controlled by

S0CON register. These modes can be support in different baud rate and communication protocols.

| SM0 | SM1 | Mode | Synchronization | Clock Rate | Start Bit | Data Bits | Stop Bit | UART pins' mode and data |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | Synchronous | Fcpu/12 | X | 8 | X | UTX pin:<br>P05M=1 and P05=1<br>URX pin:<br>Transmitter:<br>P06M=1 and P06=1<br>Receiver:<br>P06M=0 and P06=1 |
| 0 | 1 | 1 | Asynchronous | Baud rate generator or T1 overflow rate | 1 | 8 | 1 | UTX pin:<br>P05M=1 and P05=1<br>URX pin:<br>P06M=0 |
| 1 | 0 | 2 | Asynchronous | Fcpu/64 or Fcpu/32 | 1 | 9 | 1 | |
| 1 | 1 | 3 | Asynchronous | Baud rate generator or T1 overflow rate | 1 | 9 | 1 | |

## 17.2 Mode 0: Synchronous 8-bit Receiver/Transmitter

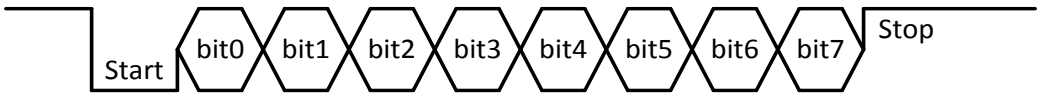
Mode0 is a shift register mode. It operates as synchronous transmitter/receiver. The UTX pin output shift clock for both transmit and receive condition. The URX pin is used to transmit and receive data. 8-bit data will be transmit and receive with LSB first. The baud rate is fcpu/12. Data transmission is started by writing data to S0BUF register. In the end of the 8th bit transmission, the TI0 flag is set. Data reception is controlled by REN0 bit and clearing RI0 bits. When REN0=1 and RI0 is from 1 to 0, data transmission starts and the RI0 flag is set at the end of the 8th bit reception.

## 17.3 Mode 1: Asynchronous 8-bit Receiver/Transmitter

Mode1 supports an asynchronous 8-bit UART with variable baud rate. The transfer format includes 1 start bit, 8 data bits (LSB first) and 1 stop bit. Data is transmitted by UTX pin and received by URX pin. The baud rate clock source can be baud rate generator or T1 overflow controlled by BD bit. When BD=0, the baud rate clock source is from T1 overflow. When BD=1, the baud rate clock source is from baud rate generator controlled by S0RELH and S0RELL. Additionally, the baud rate can be doubled by SMOD bit.

Data transmission is controlled by REN0 bit. After transmission configuration, load transmitted data into S0BUF, and then UART starts to transmit the pocket. The TI0 flag is set at the beginning of the stop bit.

Data reception is controlled by REN0 bit. When REN0=1, data reception function is enabled. Data reception starts by receiving the start bit for master terminal, URX detects the falling edge of start bit, and then the RI0 flag is set in the middle of a stop bit. Until reception completion, input data is stored in S0BUF register and the stop bit is stored in RB80.




## 17.4 Mode 2: 9-bit Receiver/Transmitter with Fixed Baud Rate

Mode2 supports an asynchronous 9-bit UART with fixed baud rate. The transfer format includes 1 start bit, 9 data bits (LSB first) and 1 stop bit. Data is transmitted by UTX pin and received by URX pin. The baud rate clock source is fixed to fcpu/64 or fcpu/32 and is controlled by SMOD bit. When SMOD=0, baud rate is fcpu/64. When SMOD=1, baud rate is fcpu/32.

Data transmission is controlled by REN0 bit. After transmission configuration, load transmitted data into S0BUF, and then UART starts to transmit the pocket. The 9th data bit is taken from TB80. The TI0 flag is set at the beginning of the stop bit.

Data reception is controlled by REN0 bit. When REN0=1, data reception function is enabled. Data reception starts by receiving the start bit for master terminal, URX detects the falling edge of start bit, and then the RI0 flag is set in the middle of a stop bit. Until reception completion, lower 8-bit input data is stored in S0BUF register and the 9th bit is stored in RB80.


## 17.5 Mode 3: 9-bit Receiver/Transmitter with Variable Baud Rate

Mode3 supports an asynchronous 9-bit UART with variable baud rate. The transfer format includes 1 start bit, 9 data bits (LSB first) and 1 stop bit. Data is transmitted by UTX pin and received by URX pin. The different between Mode2 and Mode3 is baud rate selection. In the Mode3, the baud rate clock source can be baud rate generator or T1 overflow controlled by BD bit. When BD=0, the baud rate clock source is from T1 overflow. When BD=0, the baud rate clock source is from baud rate generator controlled by S0RELH and S0RELL. Additionally, the baud rate can be doubled by SMOD bit.

Data transmission is controlled by REN0 bit. After transmission configuration, load transmitted data into S0BUF, and then UART starts to transmit the pocket. The 9th data bit is taken from TB80. The TI0 flag is set at the beginning of the stop bit.

Data reception is controlled by REN0 bit. When REN0=1, data reception function is enabled. Data reception starts by receiving the start bit for master terminal, URX0 detects the falling edge of start bit, and then the RI0 flag is set in the middle of a stop bit. Until reception completion, lower 8-bit input data is stored in S0BUF register and the 9th bit is stored in RB80.


## 17.6 Multiprocessor Communication

UART supports multiprocessor communication between a master device and one or more slaver device in Mode2 and Mode3 (9-bit UART). The master identifies correct slavers by using the 9th data bit. When the communication starts, the master transmits a specific address byte with the 9th bit is set “1” to selected slavers, and then transmits a data byte with the 9th bit is set “0” in the following transmission.

Multiprocessor communication is controlled by SM20 bit. When SM20=0, disable multiprocessor communication. When SM20=1, enable multiprocessor communication. If SM20 is set, the UART reception interrupt is only generated when the 9th received bit is “1” (RB80). The slavers will compare received data with its own address data by software. If address byte is match, the slavers clear SM20 bit to enable interrupt function in the following data transmission. The slavers with unmatched address, their SM20 keep in “1” and will not generate interrupt in the following data transmission.

## 17.7 Baud Rate Control

The UART mode 0 has a fixed baud rate at fcpu/12, and the mode 2 has two baud rate selection which is chosen by SMOD register: fcpu/64 (SMOD = 0) and fcpu/32 (SMOD = 1).

The baud rate of UART mode 1 and mode 3 is generated by either S0RELH/S0RELL registers (BD = 1) or Timer 1 overflow period (BD = 0). The SMOD bit doubles the frequency from the generator.

If the S0RELH/S0RELL is selected (BD = 1) in mode 1 and 3, the baud rate is generated as following equation.

$$\text{Baud Rate} = 2^{\text{SMOD}} \times \frac{\text{fcpu}}{64 \times (1024 - \text{S0REL})} bps$$

Table 17-1 Recommended Setting for Common UART Baud Rates (fcpu = 8 MHz)

| Baud Rate | SMOD | S0RELH | S0RELL | Accuracy |
|---|---|---|---|---|
| 4800 | 0 | 0x03 | 0xE6 | 0.16 % |
| 9600 | 0 | 0x03 | 0xF3 | 0.16 % |
| 19200 | 1 | 0x03 | 0xF3 | 0.16 % |
| 38400 | 1 | 0x03 | 0xF9 | -6.99 % |
| 56000 | 1 | 0x03 | 0xFB | -10.71 % |
| 57600 | 1 | 0x03 | 0xFC | 8.51 % |
| 115200 | 1 | 0x03 | 0xFE | 8.51 % |
| 128000 | 1 | 0x03 | 0xFE | -2.34 % |
| 250000 | 1 | 0x03 | 0xFF | 0 % |

If the Timer 1 overflow period is selected (BD = 0) in mode 1 and 3, the baud rate is generated as following equation. The Timer 1 must be in 8-bit auto-reload mode which can generate periodically overflow signals.

$$\text{Baud Rate} = 2^{\text{SMOD}} \times \frac{\text{T1 clock rate}}{32 \times (256 - \text{TH1})} bps$$

Table 17-2 Recommended Setting T1 overflow period (T1 clock=32M) for Common UART Baud Rates (fcpu = 8 MHz)

| Baud Rate | SMOD | Timer Period | TH1/TL1 | Accuracy |
|---|---|---|---|---|
| 4800 | 0 | 6.510 us | 0x30 | 0.16 % |
| 9600 | 1 | 6.510 us | 0x30 | 0.16 % |
| 19200 | 1 | 3.255 us | 0x98 | 0.16 % |
| 38400 | 1 | 1.628 us | 0xCC | 0.16 % |
| 56000 | 1 | 1.116 us | 0xDC | -0.80 % |
| 57600 | 1 | 1.085 us | 0xDD | -0.80 % |
| 115200 | 1 | 0.543 us | 0xEF | 2.08 % |
| 128000 | 1 | 0.488 us | 0xF0 | -2.40 % |

✷ ***Note:***

***1. When baud rate generator source is T1 overflow rate, the max counter value is 0xFB. (Only supports 0x00~0xFB).***

***2. When baud rate generator source is T1 overflow rate, the system clock fcpu must be greater four times to T1 overflow rate.***

## 17.8 Power Saving

The UART module has clock gating function for saving power. When REN0 bit is 0, the UART module internal clocks are halted to reduce power consumption. UART relevant register (S0CON, S0CON2, S0BUF, S0RELL, S0RELH and SMOD bit) are unable to access.

Conversely, when REN0 bit is 1, UART internal clocks are run, and registers can access. The REN0 bit must be set to 1, before the initial setting UART.

## 17.9 UART Registers

| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| S0CON | SM0 | SM1 | SM20 | REN0 | TB80 | RB80 | TI0 | RI0 |
| S0CON2 | BD | - | - | - | - | - | - | - |
| S0BUF | S0BUF7 | S0BUF6 | S0BUF5 | S0BUF4 | S0BUF3 | S0BUF2 | S0BUF1 | S0BUF0 |
| PCON | SMOD | - | - | - | P2SEL | GF0 | STOP | IDLE |
| S0RELH | - | - | - | - | - | - | S0REL9 | S0REL8 |
| S0RELL | S0REL7 | S0REL6 | S0REL5 | S0REL4 | S0REL3 | S0REL2 | S0REL1 | R0REL0 |
| IEN0 | EAL | - | ET2 | ES0 | ET1 | EX1 | ET0 | EX0 |
| P0OC | - | - | - | P15OC | P14OC | P13OC | P06OC | P05OC |
| P0M | P07M | P06M | P05M | P04M | P03M | P02M | P01M | P00M |
| P0 | P07 | P06 | P05 | P04 | P03 | P02 | P01 | P00 |

**S0CON Register (0x98)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..6 | SM[0:1] | R/W | 00 | UART mode selection<br>00: Mode 0<br>01: Mode 1<br>10: Mode 2<br>11: Mode 3 |
| 5 | SM20 | R/W | 0 | Multiprocessor communication (mode 2, 3)<br>0: Disable<br>1: Enable |
| 4 | REN0 | R/W | 0 | UART module (and reception function)<br>0: Disable for power saving*<br>1: Enable for UART operating |
| 3 | TB0 | R/W | 0 | The 9th bit transmission data (mode 2, 3) |
| 2 | RB0 | R/W | 0 | The 9th bit data from reception |

| | | | | |
|---|---|---|---|---|
| 1 | TI0 | R/W | 0 | UART interrupt flag of transmission |
| 0 | RI0 | R/W | 0 | UART interrupt flag of reception |

* When REN0 bit is 0, UART relevant register are unable to access, and the module internal clocks are halted.

✷ ***Note: TI0 and RI0 are clear by software when interrupt is enabled.***

**S0CON2 Register (0xD8)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | BD | R/W | 0 | Baud rate generators selection (mode 1, 3)<br>0: Timer 1 overflow period<br>1: Controlled by S0RELH, S0RELL registers |
| 6..0 | Reserved | R | 0x00 | |

**S0BUF Register (0x99)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | S0BUF | R/W | 0x00 | Action of writing data triggers UART communication (LSB first). Reception data is available to read by the end of packages. |

**PCON Register (0x87)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | SMOD | R/W | 0 | UART baud rate control<br>In UART mode 0: Unused.<br>In UART mode 1, 3: The baud rate is generated as the equation in section 17.7 (Baud Rate Control).<br>In UART mode 2:<br>0: fcpu/64<br>1: fcpu/32 |
| 6..0 | | | | Refer to other chapter(s) |

**IEN0 Register (0xA8)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | EAL | R/W | 0 | Interrupts enable. Refer to Chapter Interrupt |
| 4 | ES0 | R/W | 0 | Enable UART interrupt |

| | |
|---|---|
| Else | Refer to other chapter(s) |

**P0OC Register (0xE4)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 1 | P06OC | R/W | 0 | 0: Switch P0.6 (URX) to input mode (required)<br>~~1: Switch P0.6 (URX) to open-drain mode~~* |
| 0 | P05OC | R/W | 0 | 0: Switch P0.5 (UTX) to push-pull mode<br>1: Switch P0.5 (UTX) to open-drain mode |
| Else | | | | Refer to other chapter(s) |

* Setting P06OC as high causes URX cannot receive data.

**P0M Register (0xF9)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 6 | P06M | R/W | 0 | 0: Set P0.6 (URX) as input mode (required)<br>~~1: Set P0.6 (URX) as output mode~~* |
| 5 | P05M | R/W | 0 | ~~0: Set P0.5 (UTX) as input mode~~*<br>1: Set P0.5 (UTX) as output mode (required) |
| Else | | | | Refer to other chapter(s) |

* The URX and UTX respectively require input and output mode selection to receive/transmit data appropriately.

**P0 Register (0x80)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 6 | P06 | R/W | 0 | This bit is available to read at any time for monitoring the bus statue. |
| 5 | P05 | R/W | 0 | ~~0: Set P0.5 (UTX) always low~~*<br>1: Make P0.5 (UTX) can output UART data (required) |
| Else | | | | Refer to other chapter(s) |

* Setting P05 initially high because UART block drive the shared pin low signal only.

## 17.10 Sample Code

The following sample code demonstrates how to perform UART mode 1 with interrupt.

```
#define SYSUartSM0    (0 << 6)
#define SYSUartSM1    (1 << 6)
#define SYSUartSM2    (2 << 6)
#define SYSUartSM3    (3 << 6)
#define SYSUartREN    (1 << 4)
#define SYSUartSMOD   (1 << 7)
#define SYSUartES0    (1 << 4)

void SYSUartInit(void)
{
  // set UTX, URX pins' mode at here or at GPIO initialization
  P05 = 1;
  P0M = P0M | 0x20 & ~0x40;

  // configure UART mode between SM0 and SM3, enable URX
  S0CON = SYSUartSM1 | SYSUartREN;

  // configure UART baud rate
  PCON = SYSUartSMODE1;
  S0CON2 = SYSUartBD1;
  S0RELH = 0x03;
  S0RELL = 0xFE;

  // enable UART interrupt
  IEN0 |= SYSUartES0;

  // send first UTX data
  S0BUF = uartTxBuf;
}

void SYSUartInterrupt(void) interrupt ISRUart//0x23
{
  if (TI0 == 1) {
    S0BUF = uartTxBuf;
    TI0 = 0;
  } else if (RI0 == 1) {
    uartRxBuf = S0BUF;
    RI0 = 0;
  }
}
```

# 18 SPI

The SPI a serial communicate interface for data exchanging from one MCU to one MCU or other hardware peripherals. It is a simple 8-bit interface without a major definition of protocol, packet or control bits. The SPI transceiver includes three pins, clock (SCK), data input and data output (MISO/ MOSI) to send data between master and slaver terminals. An optional slave select pin (SSN) can be enabled by register in slave mode. The SPI interface builds in 4-mode which are the clock idle status and the clock phases.

- Full-duplex, 3-wire synchronous data transfer.
- Master (SCK is clock output) or Slave (SCK is clock input) operation.
- Seven SPI Master baud rates.
- Slave Clock rate up to fcpu/8.
- 8-bit data transmitted MSB first, LSB last.
- Serial clock with programmable polarity and phase.
- Master Mode fault error flag with MCU interrupt capability.
- Write collision flag protection.

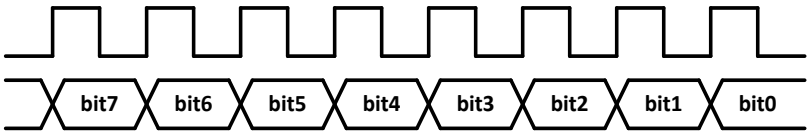
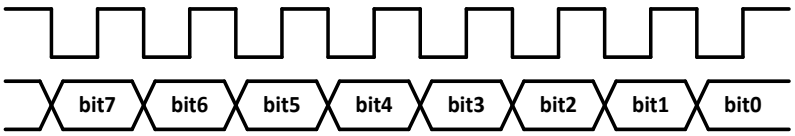
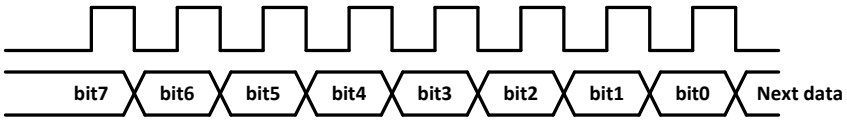
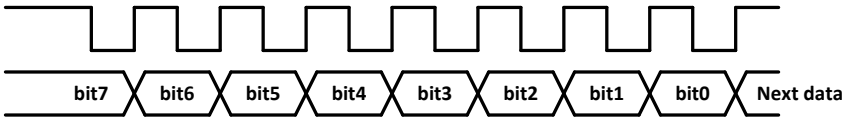
## 18.1 SPI Operation

The SPCON register can control SPI operating function, such as: transmit/receive, clock rate, data transfer direction, SPI clock idle status and clock control phase and enable this circuit. This SPI circuit will transmit or receive 8-bit data automatically by setting SPEN in SPCON register and write or read SPDAT register.

CPOL bit is designed to control SPI clock idle status. CPHA bit is designed to control the clock edge direction of data receive. CPOL and CPHA bits decide the SPI format. The SPI data transfer direction is MSB bit to LSB bit.

The SPI supports 4-mode format controlled by CPOL and CPHA bits. The edge direction is “Data Transfer Edge”. When setting rising edge that means to receive and transmit one bit data at SCK rising edge, and data transition is at SCK falling edge. When setting falling edge, that means to receive and transmit one bit data at SCK falling edge, and data transition is at SCK rising edge.

“CPHA” is the clock phase bit controls the phase of the clock on which data is sampled. When CPHA=1, the SCK first edge is for data transition, and receive and transmit data is at SCK 2nd edge. When CPHA=0, the 1st bit is fixed already, and the SCK first edge is to receive and transmit data. The SPI data transfer timing as following figure:

| CPOL | CPHA | Diagrams | Description |
|---|---|---|---|
| 0 | 1 |  | SCK idle status = Low.<br>The transfer first bit = MSB.<br>SCK data transfer edge = Falling edge. |
| 1 | 1 |  | SCK idle status = High.<br>The transfer first bit = MSB.<br>SCK data transfer edge = Rising edge. |
| 0 | 0 |  | SCK idle status = Low.<br>The transfer first bit = MSB.<br>SCK data transfer edge = Rising edge. |
| 1 | 0 |  | SCK idle status = High.<br>The transfer first bit = MSB.<br>SCK data transfer edge = Falling edge. |

The SPI supports interrupt function. ESPI is SPI interrupt function control bit. ESPI=0, disable SPI interrupt function. ESPI=1, enable SPI interrupt function. When SPI interrupt function enable, the program counter points to interrupt vector to do SPI interrupt service routine after SPI operating. SPIF is SPI interrupt request flag, and also to be the SPI operating status indicator when ESPI= 0, but cleared by reading the SPSTA, SPDAT registers.

SPI builds in chip selection function to implement SPI multi-device mode. One master communicating with several slave devices in SPI bus, and the chip selection decides the pointed device. The chip selection pin is SSN pin.

The SPI pins also support open-drain structure. The open-drain option is controlled by PnOC bits. When PnOC=0, disable SPI open-drain structure. When PnOC=1, enable SPI open-drain structure. If enable open-drain structure, SPI pins must be set input mode and need external pull-up resistor.

## 18.2 SPI Master

The SPI master mode has seven types of clock generator from fcpu/2 to fcpu/128. Generated clock is outputted through SCK pin (shared with P1.3) and its idle status is controlled by CPOL.

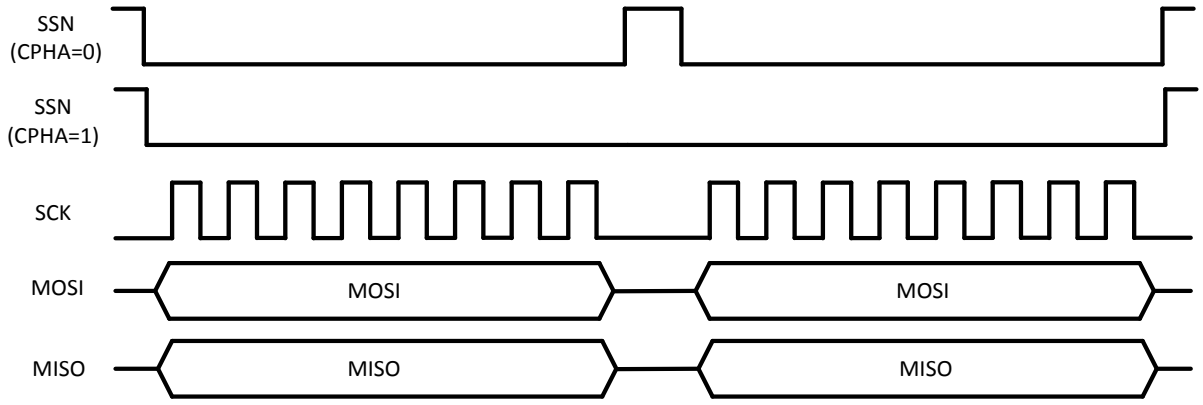
The phase of data input and output is automatically specified by CPHA register. In master mode MOSI pin (shared with P1.4) plays the role of data output, and MISO pin (shared with P1.5) fetches data from slave device. A SPI communication is started by writing SPDAT register; the received data from MISO is available to read after the end of data transmission.

The master mode has two status flags with interrupt function:
SPIF register indicates the end of one byte data communication. An interrupt would be issued at the same time if ESPI bit is enabled.
MODF is issued by SSN (shared with P0.2) low status while transmission. This interrupt source can be masked by setting SSDIS bit.

## 18.3 SPI Slave

The SPI slave mode monitors SCK pin to control its MISO and MOSI communication. However, the maximum clock rate is limited at fcpu/8. Slave device(s) are expected to specify its CPOL and CPHA setting as the same configuration of the connected SPI bus.

The slave mode treats MOSI pin as its data input, and MISO pin as its data transmission. By default, the SSDIS register is low which means the slave select pin (SSN) is functional. A SPI communication would be processed if the SSN is low status. Thus, a slave device is suspended if its SSN is high status. But in CPHA = 0, Strictly SSN must follow each 8-bit data needs to be included with falling edge and rising edge, CPHA=1 is not limitation.



The slave mode has two status flags with interrupt function:
SPIF indicates the end of one byte data communication. The original SPDAT's value has been transmitted, and the received data from MOSI is ready to be read on SPDAT.
MODF indicates that the slave select pin (SSN) has turned high before a completion of one byte communication. In other word, the last time of SPI communication is broken.

## 18.4 Power Saving

The SPI module has clock gating function for saving power. When SPEN bit is 0, the SPI module internal clocks are halted to reduce power consumption. SPI relevant register (SPCON, SPSTA and SPDAT) are unable to access. Conversely, when SPEN bit is 1, SPI internal clocks are run, and registers can access. The SPEN bit must be set to 1, before the initial setting SPI.

## 18.5 SPI Registers

| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| SPCON | SPR2 | SPEN | SSDIS | MATR | CPOL | CPHA | SPR1 | SPR0 |
| SPSTA | SPIF | WCOL | SSERR | MODF | - | - | - | - |
| SPDAT | SPDAT7 | SPDAT6 | SPDAT5 | SPDAT4 | SPDAT3 | SPDAT2 | SPDAT1 | SPDAT0 |
| IEN0 | EAL | - | ET2 | ES0 | ET1 | - | ET0 | EX0 |
| IEN1 | ET2RL | - | ET2C3 | ET2C2 | ET2C1 | ET2C0 | ESPI | EI2C |
| P0OC | - | - | - | P15OC | P14OC | P13OC | P06OC | P05OC |
| P1M | P17M | P16M | P15M | P14M | P13M | P12M | P11M | P10M |

**SPCON Register (0xE2)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7,1,0 | SM[2:0] | R/W | 000 | SPI baud rate generator (master mode only)<br>000: fcpu/2<br>001: fcpu/4<br>010: fcpu/8<br>011: fcpu/16<br>100: fcpu/32<br>101: fcpu/64<br>110: fcpu/128<br>~~111: reserved~~ |
| 6 | SPEN | R/W | 0 | SPI communication function<br>0: Disable for power saving*<br>1: Enable for SPI operating |
| 5 | SSDIS | R/W | 0 | Slave select pin function (MSTR = 0, CPHA = 0 only)<br>0: Enable slave selection pin (SSN) function<br>1: Disable slave select pin (SSN) function |
| 4 | MSTR | R/W | 1 | SPI mode<br>0: Slave mode<br>1: Master mode |
| 3 | CPOL | R/W | 0 | SCK pin idle status<br>0: SCK idle low<br>1: SCK idle high |
| 2 | CPHA | R/W | 1 | Clock phase of data latch control<br>0: Data latched by the first of clock edge<br>1: Data latched by the second of clock edge |

* When SPEN bit is 0, SPI relevant register are unable to access, and the module internal clocks are halted.

**SPSTA Register (0xE1)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | SPIF | R | 0 | SPI complete communication flag<br>Set automatically at the end of communication<br>Cleared automatically by reading SPSTA, SPDAT registers |
| 6 | WCOL | R | 0 | Write collision flag<br>Set automatically if write SPDAT during communication<br>Cleared automatically by reading SPSTA, SPDAT registers |
| 5 | SSERR | R | 0 | Synchronous slave select pin error<br>Set automatically if SSN error controlling<br>Cleared automatically by clear SPEN |
| 4 | MODF | R | 0 | Mode fault flag |
| 3..0 | Reserved | R | 0x00 | |

**SPDAT Register (0xE3)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | SPDAT | R/W | 0x00 | Master mode: action of writing data triggers SPI communication; reception data is readable after the end of one byte communication (SPIF automatically set).<br>Slave mode: written data would be transmitted by SCK input; reception data is available to read after the end of one byte communication (SPIF automatically set). |

**IEN0 Register (0xA8)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | EAL | R/W | 0 | Interrupts enable. Refer to Chapter Interrupt |
| Else | | | | Refer to other chapter(s) |

**IEN1 Register (0xB8)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 1 | ESPI | R/W | 0 | Enable SPI interrupt |
| Else | | | | Refer to other chapter(s) |

**P0OC Register (0xE4)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 4 | P15OC | R/W | 0 | 0: Switch P1.5 (MISO) to input or output mode<br>1: Switch P1.5 (MISO) to open-drain mod |
| 3 | P14OC | R/W | 0 | 0: Switch P1.4 (MOSI) to input or output mode<br>1: Switch P1.4 (MOSI) to open-drain mode |
| 2 | P13OC | R/W | 0 | 0: Switch P1.3 (SCK) to input or output mode<br>1: Switch P1.3 (SCK) to open-drain mod |
| Else | | | | Refer to other chapter(s) |

**P1M Register (0xFA)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 5 | P15M | R/W | 0 | 0: Set P1.5 (MISO) as input mode [slave mode]<br>1: Set P1.5 (MISO) as output mode [master mode] |
| 4 | P14M | R/W | 0 | 0: Set P1.4 (MOSI) as input mode [master mode]<br>1: Set P1.4 (MOSI) as output mode [slave mode] |
| 3 | P13M | R/W | 0 | 0: Set P1.3 (SCK) as input mode [slave mode]<br>1: Set P1.3 (SCK) as output mode [master mode] |
| Else | | | | Refer to other chapter(s) |

[1]Setting SCK as input mode is essential in slave mode; setting as output mode is recommended in master mode.

[2]Setting MISO as input mode is essential in master mode; setting as output mode is recommended in slave mode.

[3]Setting MOSI as input mode is essential is slave mode; setting as output mode is recommended in master mode.

**P0M Register (0xF9)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 2 | P02M | R/W | 0 | 0: Set P0.2 (SSN) as input mode*<br>1: Set P0.2 (SSN) as output mode* |

*If slave mode with SSN function: essentially to set SSN as input mode.

## 18.6 Sample Code

The following sample code demonstrates how to perform SPI Master with interrupt.

```
#define SpiMaster      (1 << 4)  //SPI = Master mode
#define SpiSlave       (1 << 4)  //SPI = Slave mode
#define SpiMode0       (0 << 2)  //SCK idle low, data latch at rising edge
#define SpiMode1       (1 << 2)  //SCK idle low, data latch at falling edge
#define SpiMode2       (2 << 2)  //SCK idle high, data latch at falling edge
#define SpiMode3       (3 << 2)  //SCK idle high, data latch at rising edge
#define SpiEn          (1 << 6)  //Enable SPI
#define SpiSSNEn       (0 << 5)  //SSN pin function enable
#define SpiSSNDis      (1 << 5)  //SSN pin function disable

unsigned char  u8SpiData = 0;    // data buffer
unsigned char  u8TxCompleted = 0;

void SpiMaster(void)
{
  unsigned char u8RcvData = 0;

  //SCK & MOSI = output, MISO = input
  P1M |= 0x18;
  //Enable Spi, Master mode, SSN pin disable, Fclk/128
  //SCK idle low, data latch at falling edge
  SPCON = SpiEn | SpiMaster | SpiMode1 | SpiSSNDis | 0x82;
  //Enable Global/SPI interrupt
  IEN1  |= 0x02;
  IEN0  |= 0x80;    //enable global interrupt

  while (1) {
    SPDAT = 0x55;
    while(!u8TxCompleted);     // wait end of transmition
    u8TxCompleted = 0;         // clear sw flag
    u8RcvData = u8SpiData;     // receive 0x66

    SPDAT = 0x99;
    while(!u8TxCompleted);     // wait end of transmition
    u8TxCompleted = 0;         // clear sw flag
    u8RcvData = u8SpiData;     // receive 0xAA
  }
}

void SpiInterrupt(void) interrupt ISRSpi //0x4B
{
  switch ( SPSTA )             // Clear SPI flag (SPIF) by reading
  {
    case 0x80:
      u8SpiData = SPDAT;
      u8TxCompleted = 1;
      break;
    case 0x10:
      // Mode Fault
      break;
  }
}
```

# 19 I2C

The I2C is a serial communication interface for data exchanging from one MCU to one MCU or other hardware peripherals. The device can transmit data as a master or a slave with two bi-directional IO, SDA (Serial data output) and SCL (Serial clock input).

When a master transmit data to a slave, it's called "WRITE" operation; when a slave transmit data to a master, it's called "READ" operation. It also supports multi-master communication and keeps data transmission correctly by an arbitration method to decide one master has the control on bus and transmit its data.

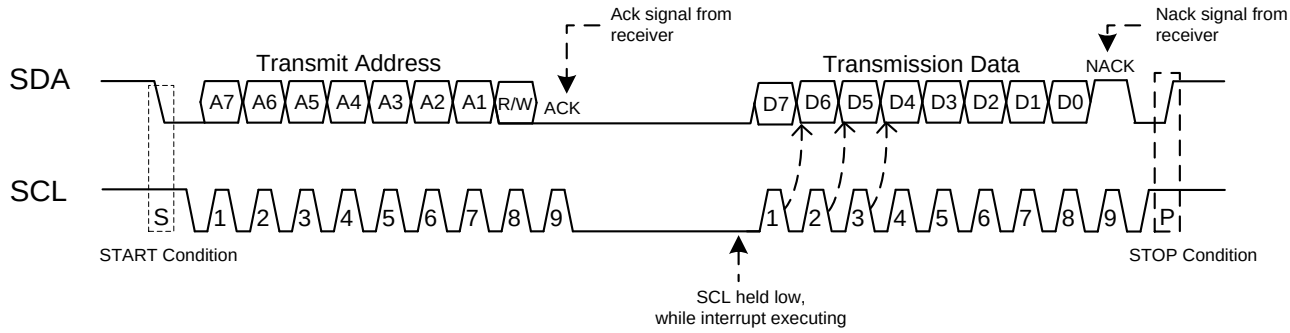
- Master Tx, Rx Mode
- Slave Tx, Rx mode (with general address call) for multiplex slave in single master situation.
- 2-wire synchronous data transfer/receiver.
- Support 100K/400K clock rate.

## 19.1 I2C Protocol

I2C transmission structure includes a START(S) condition, 8-bit address byte, one or more data byte and a STOP (P) condition. START condition is generated by master to initial any transmission.

Data is transmitted with the Most Significant Bit (MSB) first. In address byte, the higher 7-bit is address bit and the lowest bit is data direction (R/W) bit. When R/W=0, it assigns a "WRITR" operation. When R/W=1, it assigns a "READ" operation.

After each byte is received, the receiver (a master or a slave) must send an acknowledge (ACK). If transmitter can't receive an ACK, it will recognize a not acknowledge (NACK). In WRITE operation, the master will transmit data to the slave and then waits for ACK from slave. In READ operation, the slave will transmit data to the master and then waits for ACK from master. In the end, the master will generate a STOP condition to finish transmission.

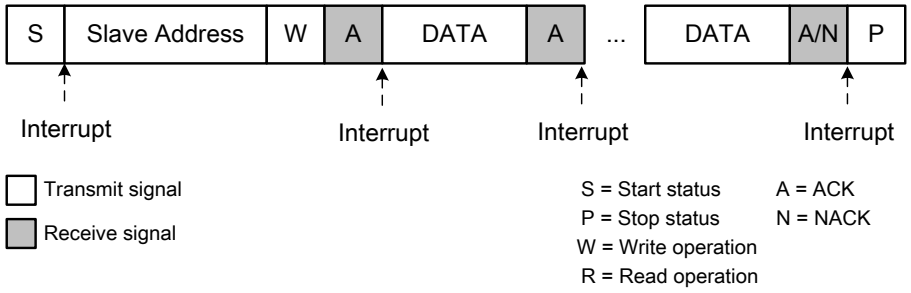


## 19.2 I2C Transfer Modes

The I2C can operate as a master/slave to execute the 8-bit serial data transmission/reception operation. Thus, the module can operate in one of four modes: Master Transmitter, Master Receiver, Slave Transmitter and Slave Receiver.

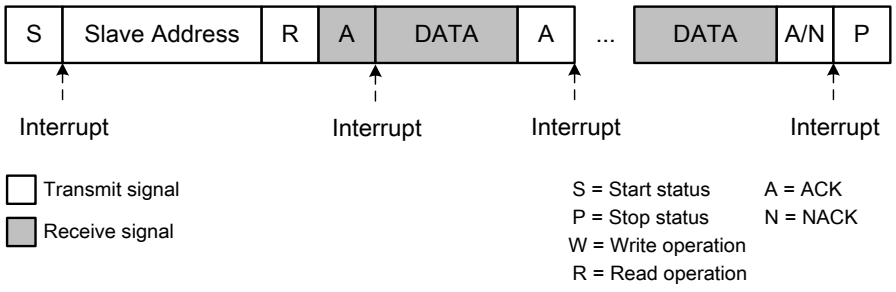
### 19.2.1 Master Transmitter Mode

The master transmits information to the slave. The serial data is output via SDA while the serial clock is output on SCL. Data transmission starts via generate a START(S) signal. After the START signal, the specific address byte of slave device is sent. The address byte includes 7-bit address bit and an 8th data direction (R/W) bit. The R/W is set “0” to enable the master transmission. In the following, the master transmits one or more data byte to the slaver. After each data is transmitted, the master waits for the acknowledge (ACK) from the slave. In the end, the master generates a STOP (P) signal to terminate the data transmission.



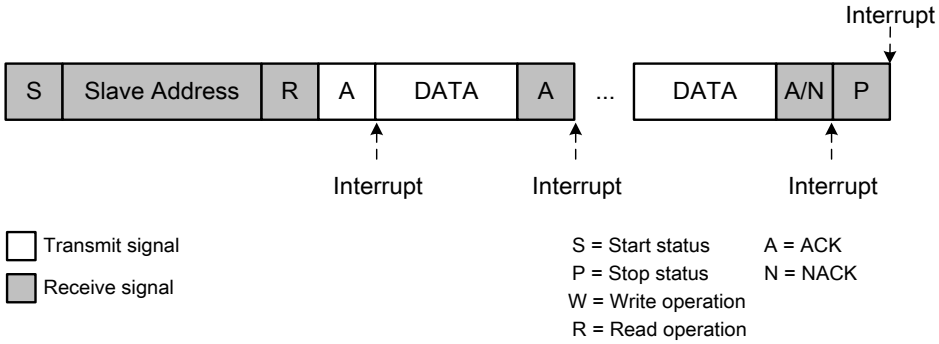

### 19.2.2 Master Receiver Mode

The master receives the information from the slave. The serial data input via SDA while the serial clock output on SCL. Data reception starts via generate a START(S) signal. After the START signal, the specific address byte of slave device is sent. The address byte includes 7-bit address bit and an 8th data direction (R/W) bit. The R/W is set “1” to enable the master reception. In the following, the master receives one or more data byte from the slaver. After each data is received, the master generates the acknowledge (ACK) or not acknowledge (NACK) to the slave via the status of AA bit. In the end, the master generates a STOP (P) signal to terminate the data transmission.



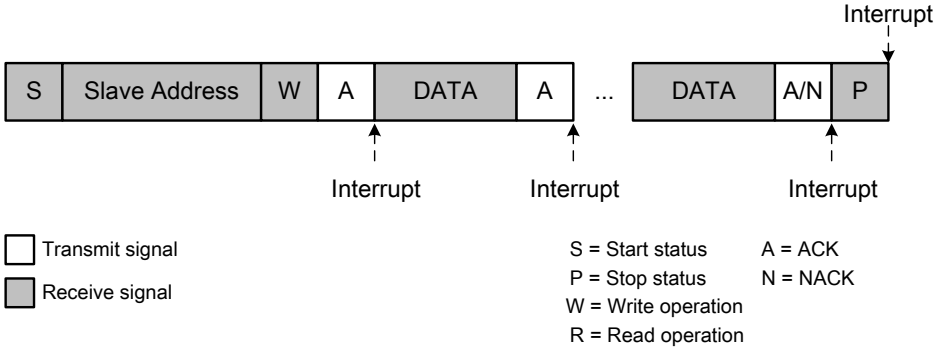
### 19.2.3 Slave Transmitter Mode

The slave transmits information to the master. The serial data output via SDA while the serial clock input on SCL. Data transmission starts via receive a START(S) signal from the master. After the START signal, the specific address byte of slave device is received. The address byte includes 7-bit address bit and an 8th data direction (R/W) bit. The R/W is set “1” to enable the slave transmission. If the received address byte match the address in I2CADR register, the slave generate an acknowledge (ACK). Otherwise, if general call address condition is set (GC=1), the slave also generate an acknowledge (ACK) after general call address (0x00) is received. In the following, the slave transmits one or more data byte to the master. After each data is transmitted, the slave waits for the acknowledge (ACK) from the master. In the end, the slave receives a STOP (P) signal from the master to terminate the data transmission.



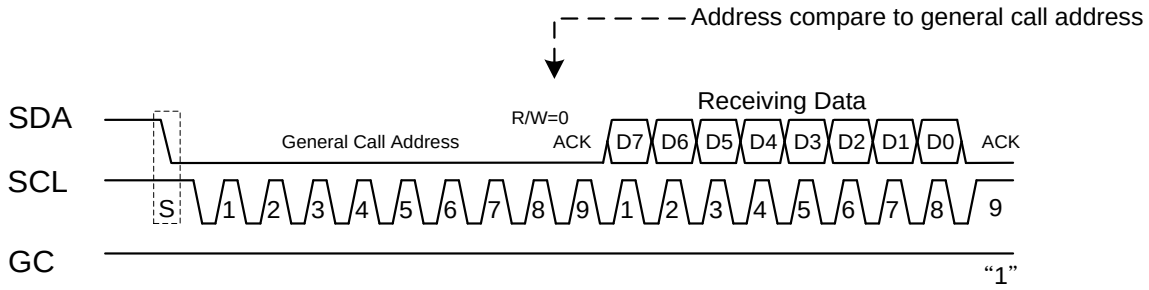

### 19.2.4 Slave Receiver Mode

The slave receives information from the master. Both the serial data and the serial clock are input on SDA and SCL. Data reception starts via receive a START(S) signal from the master. After the START signal, the specific address byte of slave device is received. The address byte includes 7-bit address bit and an 8th data direction (R/W) bit. The R/W is set “0” to enable the slave reception. If the received address byte match the address in I2CADR register, the slave generate an acknowledge (ACK). Otherwise, if general call address condition is set (GC=1), the slave also generate an acknowledge (ACK) after general call address (0x00) is received. In the following, the slave receives one or more data byte from the master. After each data is receives, the slave generates the acknowledge (ACK) or not acknowledge (NACK) to the master via the status of AA bit. In the end, the slave receives a STOP (P) signal from the master to terminate the data transmission.



## 19.3 General Call Address

In I2C bus, the first 7-bit is the slave address. Only the address matches slave address, the slave will response an ACK. The exception is the general call address which can address all slave devices. When this address occur, all devices should response an acknowledge (ACK). The general call address is a special address which is reserved as all "0" of 7-bit address. The general call address function is control by GC bit. Set this bit will enable general call address and clear it will disable. When GC=1, the general call address will be recognized. When GC=0, the general call address will be ignored.




## 19.4 Serial Clock Generator

In master mode, the SCL clock rate generator's is controlled by CR[2:0] bit of I2CCON register.

When CR[2:0]=000~110, SCL clock rate is from internal clock generator.

$$\text{SCL Clock Rate} = \frac{\text{Fcpu}}{\text{Prescaler}} (\text{Prescaler} = 256 \sim 60)$$

When CR[2:0]=111, SCL clock rate is from Timer 1 overflow rate .

$$\text{SCL Clock Rate} = \frac{\text{Timer 1 Overflow}}{8}$$

The table below shows the clock rate under different setting.

| CR2 | CR1 | CR0 | I2C | Bit Frequency (kHz) | |
|---|---|---|---|---|---|
| | | | Prescaler | 6MHz | 8MHz |
| 0 | 0 | 0 | 256 | 23 | 31 |
| 0 | 0 | 1 | 224 | 27 | 36 |
| 0 | 1 | 0 | 192 | 31 | 42 |
| 0 | 1 | 1 | 160 | 37 | 50 |
| 1 | 0 | 0 | 960 | 6.25 | 8 |
| 1 | 0 | 1 | 120 | 50 | 67 |
| 1 | 1 | 0 | 60 | 100 | 133 |
| 1 | 1 | 1 | (Timer 1 overflow rate)/8 | | |

✷ *Note:*

***1. The first step of I2C operation is to setup the I2C pins' mode. Must be set "input mode" in SDA/SCL pins.***

***2. When clock generator source is T1 overflow rate, the max counter value is 0xFB. (Only supports 0x00~0xFB). And in this time if T1 clock rate is IHRC_32MHz, SCL maximum clock rate is 800kHz.***

***3. If user wants to generate SCL clock rate is 100kHz/400kHz, you can set T1 counter value is 0xD8/0xF6 easily.***

## 19.5 Synchronization and Arbitration

In multi-master condition, more than one master may transmit on bus in the same time. It must be decided which master has the control of bus and complete its transmission. Clock synchronization and arbitration are used to configure multi-master transmission. Clock synchronization is executed by synchronizing the SCL signal with anther devices.

When two masters want to transmit data in the same, the clock synchronization will start by the High to Low transition on the SCL. If master 1 clock set LOW first, it holds the SCL in LOW status until the clock transit to HIGH status. However, if anther master clock still keep LOW status, the Low to High transition of master 1 may not change SCL status (SCL keep LOW). In the other word, SCL keep LOW by the master with the longest clock time in LOW status. The SCL will transit from LOW to HIGH when the all devices clock transit to HIGH status. In the duration, the master1 will keep in HIGH status and wait for SCL transition (from LOW to HIGH), then continue its transmission. After clock synchronization, all devices clock and SCL clock are the same. Arbitration is used to decide which master can complete its transmission by SDA signal. Two masters may send out a START condition and transmit data on bus in the same time. They may influence by each other. Arbitration will force one master to lose the control on bus. Data transmission will keep until master output different data signal. If one master transmits HIGH status and anther master transmits LOW status, the SDA will be pull low. The master output High will detect the different with SDA and lose the control on bus. The mater with LOW status wins the bus control and continues its transmission. There is no data miss during arbitration.

## 19.6 System Management Bus Extension

The optional System Management Bus (SMBus) protocol hardware supports 3 types timeout detection: (1) Tmext Timeout Detection: The cumulative stretch clock cycles within one byte. (2)Tsext Timeout Detection: The cumulative stretch clock cycles between start and stop condition. (3)Timeout Detection: The clock low measurement.

Timeout detection is controlled by SMBSEL and SMBDST registers. The SMBEXE bit of SMBSEL is SMBus extension function enable bit. When SMBEXE=1, SMBus extension function is enabled. Otherwise, Disable SMBus extension function. Timeout type and period setting is controlled by SMBTOP[2:0] and SMBDST. The period of SMBus timeout is controlled by three 16-bit buffers of Tmex, Tsext and Tout. The equation is as following.

$$\text{Tmext/Tsext/Tout} = \frac{\text{Timeout Period(sec)xFcpu(Hz)}}{1024}$$

Tmext is support by two 8-bit register of Tmext_L and Tmext_H . Tmext_L hold the low byte and Tmext_H hold high byte. Tsext is support by two 8-bit register of Tsext_L and Tsext_H . Tsext_L hold the low byte and Tsext_H hold high byte. Tout is support by two 8-bit register of Tout_L and Tout_H . Tout_L hold the low byte and Tout_H hold high byte.

| Type | Time out period | Fcpu=8MHz | |
|---|---|---|---|
| | | DEC | HEX |
| Tmext | 5ms | 39 | 27 |
| Tsext | 25ms | 195 | C3 |
| Tout | 35ms | 273 | 111 |

By the setting of SMBTOP[2:0] to choose register type (as the table below), and write to register by write data to SMBDST register.

| SMBTOP[2:0] | SMBDST | Description |
|---|---|---|
| 000 | Tmext_L | Select the low byte of Tmext register. |
| 001 | Tmext_H | Select the high byte of Tmext register. |
| 010 | Tsext_L | Select the low byte of Tsext register. |
| 011 | Tsext_H | Select the high byte of Tsext register. |
| 100 | Tout_L | Select the low byte of Tout register. |
| 101 | Tout_H | Select the high byte of Tout register. |

When the SMBus extension function is enabled the lower 3-bit of I2CSTA hold the information about time out as the table below.

| I2CSTA | Description |
|---|---|
| XXXX X000 | No timeout errors. |
| XXXX XXX1 | Tout timeout error. |
| XXXX XX1X | Tsext timeout error. |
| XXXX X1XX | Tmext timeout error. |

## 19.7 Power Saving

The I2C module has clock gating function for saving power. When ENS1 bit is 0, the I2C module internal clocks are halted to reduce power consumption. I2C relevant register (I2CDAT, I2CADR, I2CCON, I2CSTA, SMBSEL and SMBDST) are unable to access. Conversely, when ENS1 bit is 1, I2C internal clocks are run, and registers can access. The ENS1 bit must be set to 1, before the initial setting I2C.

## 19.8 I2C Registers

| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| I2CDAT | I2CDAT7 | I2CDAT6 | I2CDAT5 | I2CDAT4 | I2CDAT3 | I2CDAT2 | I2CDAT1 | I2CDAT0 |
| I2CADR | ADR6 | ADR5 | ADR4 | ADR3 | ADR2 | ADR1 | ADR0 | GC |
| I2CCON | CR2 | ENS1 | STA | STO | SI | AA | CR1 | CR0 |
| I2CSTA | I2CSTA7 | I2CSTA6 | I2CSTA5 | I2CSTA4 | I2CSTA3 | I2CSTA2 | I2CSTA1 | I2CSTA0 |
| SMBSEL | SMBEXE | - | - | - | - | SMBSTP2 | SMBSTP1 | SMBSTP0 |
| SMBDST | SMBD7 | SMBD6 | SMBD5 | SMBD4 | SMBD3 | SMBD2 | SMBD1 | SMBD0 |
| IEN0 | EAL | - | ET2 | ES0 | ET1 | - | ET0 | EX0 |
| IEN1 | ET2RL | - | ET2C3 | ET2C2 | ET2C1 | ET2C0 | ESPI | EI2C |
| P0M | P07M | P16M | P05M | P04M | P03M | P02M | P01M | P00M |
| P0 | P07 | P16 | P05 | P04 | P03 | P02 | P01 | P00 |

**I2CDAT Register (0xDA)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7:0 | I2CDAT[7:0] | R/W | 0x00 | The I2CDAT register contains a byte to be transmitted through I2C bus or a byte which has just been received through I2C bus. The CPU can read from and write to this 8-bit, directly addressable SFR while it is not in the process of byte shifting. The I2CDAT register is not |

| | | | | |
|---|---|---|---|---|
| | | | | shadowed or double buffered so the user should only read I2CDAT when an I2C interrupt occurs. |

**I2CADR Register (0xDB)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7:1 | I2CADR[6:0] | R/W | 0x00 | I2C slave address |
| 0 | GC | R/W | 0 | General call address (0X00) acknowledgment<br>0: ignored<br>1: recognized |

**I2CCON Register (0xDC)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7,1,0 | CR[2:0] | R/W | 0 | I2C clock rate<br>000: fcpu/256<br>001: fcpu/224<br>010: fcpu/192<br>011: fcpu/160<br>100: fcpu/960<br>101: fcpu/120<br>110: fcpu/60<br>111: Timer 1 overflow-period/8 |
| 6 | ENS1 | R/W | 0 | I2C functionality<br>0: Disable for power saving*<br>1: Enable for I2C operating |
| 5 | STA | R/W | 0 | START flag<br>0: No START condition is transmitted.<br>1: A START condition is transmitted if the bus is free. |
| 4 | STO | R/W | 0 | STOP flag<br>0: No STOP condition is transmitted.<br>1: A STOP condition is transmitted to the I2C bus in master mode. |
| 3 | SI | R/W | 0 | Serial interrupt flag<br>The SI is set by hardware when one of 25 out of 26 possible I2C states is entered. The only state that does not set the SI is state F8h, which indicates that no relevant state information is available. The SI flag must |

| | | | | |
|---|---|---|---|---|
| | | | | be cleared by software. In order to clear the SI bit, '0' must be written to this bit. Writing a '1' to SI bit does not change value of the SI. |
| 2 | AA | R/W | 0 | Assert acknowledge flag<br>0: A NACK will be returned when a byte has received<br>1: An ACK will be returned when a byte has received |

* When ENS1 bit is 0, I2C relevant register are unable to access, and the module internal clocks are halted.

**I2CSTA Register (0xDD)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7:3 | I2CSTA[7:3] | R | 11111 | I2C Status Code |
| 2..0 | I2CSTA[2:0] | R | 000 | SMBus Status Code |

## I2C status code and status

| Mode | Status Code | Status of the I2C | Application software response | | | | | Next action taken by I2C hardware |
|---|---|---|---|---|---|---|---|---|
| | | | To/from I2CDAT | TO I2CCON | | | | |
| | | | | STA | STO | SI | AA | |
| Master Transmitter/ Receiver | 08H | A START condition has been transmitted | Load SLA+R | X | 0 | 0 | X | SLA+R/W will be transmitted; ACK will be received |
| | 10H | A repeated START condition has been transmitted. | Load SLA+R | X | 0 | 0 | X | SLA+R/W will be transmitted; ACK will be received |
| | | | Load SLA+W | | | | | SLA+W will be transmitted; I2C will be switched to MST/TRX mode. |
| Master Transmitter | 18H | SLA+W has been transmitted; ACK has been received | Load data byte | 0 | 0 | 0 | X | Data byte will be transmitted; ACK will be received. |
| | | | No action | 1 | 0 | 0 | X | Repeated START will be transmitted. |
| | | | No action | 0 | 1 | 0 | X | STOP condition will be transmitted; STO flag will be reset. |
| | | | No action | 1 | 1 | 0 | X | STOP condition followed by a START condition will be transmitted; STO flag will be reset. |
| | 20H | SLA+W has been transmitted; not ACK has been received | Load data byte* | 0 | 0 | 0 | X | Data byte will be transmitted; ACK will be received. |
| | | | No action | 1 | 0 | 0 | X | Repeated START will be transmitted. |
| | | | No action | 0 | 1 | 0 | X | STOP condition will be transmitted; STO flag will be reset. |
| | | | No action | 1 | 1 | 0 | X | STOP condition followed by a START condition will be transmitted; STO flag will be reset. |
| | 28H | Data byte in I2CDAT has been transmitted; ACK has been received | Load data byte | 0 | 0 | 0 | X | Data byte will be transmitted; ACK bit will be received. |
| | | | No action | 1 | 0 | 0 | X | Repeated START will be transmitted. |
| | | | No action | 0 | 1 | 0 | X | STOP condition will be transmitted; STO flag will be reset. |
| | | | No action | 1 | 1 | 0 | X | STOP condition followed by a START condition will be transmitted; STO flag will be reset. |
| | 30H | Data byte in I2CDAT has been transmitted; not ACK has been received | Load data byte* | 0 | 0 | 0 | X | Data byte will be transmitted; ACK will be received. |
| | | | No action | 1 | 0 | 0 | X | Repeated START will be transmitted. |
| | | | No action | 0 | 1 | 0 | X | STOP condition will be transmitted; STO flag will be reset. |
| | | | No action | 1 | 1 | 0 | X | STOP condition followed by a START condition will be transmitted; STO flag will be reset. |
| Master Receiver | 40H | SLA+R has been transmitted; ACK has been received | No action | 0 | 0 | 0 | 0 | Data byte will be received; not ACK will be returned |
| | | | No action | 0 | 0 | 0 | 1 | Data byte will be received; ACK will be returned |
| | 48H | SLA+R has been transmitted; not ACK has been received | No action | 1 | 0 | 0 | X | Repeated START condition will be transmitted |
| | | | No action | 0 | 1 | 0 | X | STOP condition will be transmitted; STO flag will be reset |
| | | | No action | 1 | 1 | 0 | X | STOP condition followed by a START condition will be transmitted; STO flag will be reset |
| | 50H | Data byte has been received; ACK has been returned | Read data byte | 0 | 0 | 0 | 0 | Data byte will be received; not ACK will be returned |
| | | | Read data byte | 0 | 0 | 0 | 1 | Data byte will be received; ACK will be returned |
| | 58H | Data byte has been received; not ACK has been returned | Read data byte | 1 | 0 | 0 | X | Repeated START condition will be transmitted |
| | | | Read data byte | 0 | 1 | 0 | X | STOP condition will be transmitted; STO flag will be reset |
| | | | Read data byte | 1 | 1 | 0 | X | STOP condition followed by a START condition will be transmitted; STO flag will be reset |

| Mode | Status Code | Status of the I2C | Application software response | | | | | Next action taken by I2C hardware |
|---|---|---|---|---|---|---|---|---|
| | | | To/from I2CDAT | TO I2CCON | | | | |
| | | | | STA | STO | SI | AA | |
| Slave Receiver | 60H | Own SLA+W has been received; ACK has been returned | No action | X | 0 | 0 | 0/1 | Data byte will be received and not ACK/ACK will be returned |
| | 68H | Arbitration lost in SLA+R/W as master; own SLA+W has been received, ACK returned | No action | X | 0 | 0 | 0/1 | Data byte will be received and not ACK/ACK will be returned |
| | 70H | General call address (00H) has been received; ACK has been returned | No action | X | 0 | 0 | 0/1 | Data byte will be received and not ACK/ACK will be returned |
| | 78H | Arbitration lost in SLA+R/W as master; general call address has been received, ACK returned | No action | X | 0 | 0 | 0/1 | Data byte will be received and not ACK/ACK will be returned |
| | 80H | Previously addressed with own SLV address; DATA has been received; ACK returned | Read data byte | X | 0 | 0 | 0/1 | Data byte will be received and not ACK/ACK will be returned |
| | 88H | Previously addressed with own SLA; DATA byte has been received; not ACK returned | Read data byte | 0 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or general call address |
| | | | Read data byte | 0 | 0 | 0 | 1 | Switched to not addressed SLV mode; own SLA or general call address will be recognized |
| | | | Read data byte | 1 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or general call address; START condition will be transmitted when the bus becomes free |
| | | | Read data byte | 1 | 0 | 0 | 1 | Switched to not addressed SLV mode; own SLA or general |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | call address will be recognized; START condition will be transmitted when the bus becomes free |
| | 90H | Previously addressed with general call address; DATA has been received; ACK returned | Read data byte | X | 0 | 0 | 0/1 | Data byte will be received and not ACK/ACK will be returned |
| | 98H | Previously addressed with general call address; DATA has been received; not ACK returned | Read data byte | 0 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or general call address |
| | | | Read data byte | 0 | 0 | 0 | 1 | Switched to not addressed SLV mode; own SLA or general call address will be recognized |
| | | | Read data byte | 1 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or general call address; START condition will be transmitted when the bus becomes free |
| | | | Read data byte | 1 | 0 | 0 | 1 | Switched to not addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free |
| | A0H | A STOP condition or repeated START condition has been received while still addressedas SLV/REC or SLV/TRX | No action | 0 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or general call address |
| | | | No action | 0 | 0 | 0 | 1 | Switched to not addressed SLV mode; own SLA or general call address will be recognized |
| | | | No action | 1 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or general call address; START condition will be transmitted when the bus becomes free |
| | | | No action | 1 | 0 | 0 | 1 | Switched to not addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free |
| Slave Transmitter | A8H | Own SLA+R has been received; ACK has been returned | Load data byte | X | 0 | 0 | 0 | Last data byte will be transmitted and ACK will be received |
| | | | Load data byte | X | 0 | 0 | 1 | Data byte will be transmitted; ACK will be received. |
| | B0H | Arbitration lost in SLA+R/W as master; own SLA+R has been received, ACK has been returned. | Load data byte | X | 0 | 0 | 0 | Last data byte will be transmitted and ACK will be received |
| | | | Load data byte | X | 0 | 0 | 1 | Data byte will be transmitted; ACK will be received. |
| | B8H | Data byte has been transmitted; ACK will be received. | Load data byte | X | 0 | 0 | 0 | Last data byte will be transmitted and ACK will be received |
| | | | Load data byte | X | 0 | 0 | 1 | Data byte will be transmitted; ACK will be received. |
| | C0H | Data byte has been transmitted; not ACK has been received. | No action | 0 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or general call address. |
| | | | No action | 0 | 0 | 0 | 1 | Switched to not addressed SLV mode; own SLA or general call address will be recognized. |
| | | | No action | 1 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or general call address; START condition will be transmitted when the bus becomes free. |
| | | | No action | 1 | 0 | 0 | 1 | Switched to not addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free. |
| | C8H | Last data byte has been transmitted; ACK has been received. | No action | 0 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or general call address. |
| | | | No action | 0 | 0 | 0 | 1 | Switched to not addressed SLV mode; own SLA or general call address will be recognized. |
| | | | No action | 1 | 0 | 0 | 0 | Switched to not addressed SLV mode; no recognition of own SLA or general call address; START condition will be transmitted when the bus becomes free. |
| | | | No action | 1 | 0 | 0 | 1 | Switched to not addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free. |
| Miscellaneous | F8H | No relevant state information available; SI=0 | No action | No action | | | | Wait or proceed current transfer |
| | 38H | Arbitration lost | No action | 0 | 0 | 0 | X | I2C will be released; A start condition will be transmitted. |
| | | | No action | 1 | 0 | 0 | X | When the bus becomes free. (enter to a master mode) |
| | 00H | Bus error during MST or selected slave modes | No action | 0 | 1 | 0 | X | Only the internal hardware is affected in the MST or addressed SLV modes. In all cases, the bus is released and I2C is switched to the not addressed SLV mode. STO flag is reset. |

"SLA" means slave address, "R" means R/W=1, "W" means R/W=0

*For applications where NACK doesn't mean the end of communication.

**SMBSEL Register (0xDE)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | SMBEXE | R/W | 0 | SMBus extension functionality<br>0: Disable<br>1: Enable |
| 2..0 | SMBSTP[2:0] | R/W | 000 | SMBus timeout register |

**SMBDST Register (0xDF)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | SMBD[7:0] | R/W | 0x00 | This register is used to provide a read/write access port to the SMBus timeout registers. Data read or written to that register is actually read or written to the Timeout Register which is pointed by the SMBSEL register. |

**IEN0 Register (0xA8)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7 | EAL | R/W | 0 | Interrupts enable. Refer to Chapter Interrupt |
| Else | | | | Refer to other chapter(s) |

**IEN1 Register (0xB8)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 0 | EI2C | R/W | 0 | Interrupts enable. Refer to Chapter Interrupt |
| Else | | | | Refer to other chapter(s) |

**P0M Register (0Xf9)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 4 | P04M | R/W | 0 | 0: Set P0.4 (SDA) as input mode (required)<br>~~1: Set P0.4 (SDA) as output mode~~* |
| 3 | P03M | R/W | 0 | 0: Set P0.3 (SCL) as input mode (required)<br>~~1: Set P0.3 (SCL) as output mode~~* |
| Else | | | | Refer to other chapter(s) |

* The P03M and P04M require be set input mode.

## 19.9 Sample Code

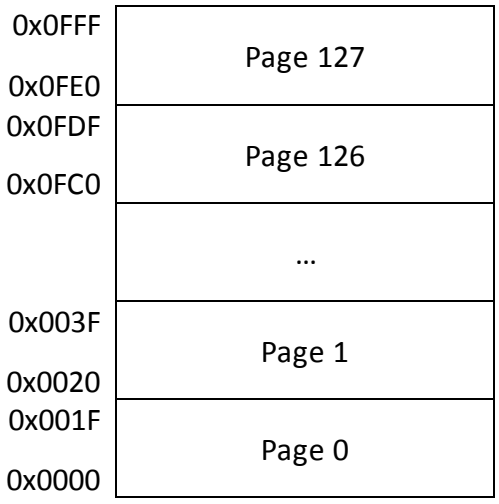
The following sample code demonstrates how to perform I2C with interrupt.

```
unsigned int  I2CAddr;
unsigned int  I2C_TXData0;
unsigned int  I2C_TXDatan;
unsigned int  I2C_RXData0;
unsigned int  I2C_RXDatan;

void I2CInit(void)
{
  P0M &= 0xE7;      // P03 & P04 as input

  // configure I2C clock(T1)and enable I2C.
  I2CCON |= 0xC3;
  TMOD = 0x60;   // auto reload
  TCON0 = 0x07;  // Fosc/1
  TH1 = 0xF6;    //400kHz
  TL1 = 0xF6;    //400kHz or
  TH1 = 0xD8;    //100kHz
  TL1 = 0xD8;    //100kHz
  TR1 = 1;

  // enable I2C interrupt
  EI2C = 1;
  //enable global interrupt
  EAL = 1;

  I2CCON |= 0x20;            // START (STA) = 1
}

void I2cInterrupt(void) interrupt ISRI2c  //0x43
{
    switch (I2CSTA)
    {
        // tx mode
        case 0x08:
            I2CCON &= 0xDF;           // START (STA) = 0
            I2CDAT = I2CAddr;        // Tx/Rx addr
            break;
        case 0x18:                    // write first byte
            I2CDAT = I2C_TXData0;
            break;
        case 0x28:                    // write n byte
            I2CDAT = I2C_TXDatan;
            break;
        case 0x30:                    // STOP (STO)
            I2CCON |= 0x10;
            break;
        //    rx mode
        case 0x40:                    // get slave addr
            I2CCON |= 0x04;           // AA = 1
            break;
        case 0x50:                    // read n byte
            I2C_RXData0 = I2CDAT;
            I2CCON &= 0xFB;           // AA = 0
            break;
```

```
54          case 0x58:                      // read last byte & stop
55              I2C_RXDatan = I2CDAT;
56              I2CCON |= 0x10;             // STOP (STO)
57              break;
58          default:
59              I2CCON |= 0x10;              // STOP (STO)
60      }
61
62      I2CCON &= 0xF7;                      // Clear I2C flag (SI)
63  }
```

# 20 In-System Program

SN8F5702 builds in an on-chip 4 KB program memory, aka IROM, which is equally divided to 128 pages (32 bytes per page). The in-system program is a procedure that enables a firmware to freely modify every page's data; in other word, it is the channel to store value(s) into the non-volatile memory and/or live update firmware.




Program memory (IROM)

## 20.1 Page Program

Because each page of the program memory has 32 bytes in length, a page program procedure requires 32 bytes IRAM as its data buffer.

| ISP ROM MAP | | ROM address bit0~bit4 (hex) =0 |
|---|---|---|
| ROM address bit5~bit15 (hex) | 0000 | These pages include reset vector and interrupt sector. We strongly recommend to reserve the area not to do ISP erase. |
| | 0020 | |
| | 0040 | |
| | ... | |
| | 00C0 | |
| | 00E0 | |
| | 0100 | One ISP Program Page |
| | 0120 | One ISP Program Page |
| | ... | One ISP Program Page |
| | 0400 | One ISP Program Page |
| | 0420 | One ISP Program Page |
| | ... | One ISP Program Page |
| | 0700 | One ISP Program Page |
| | 0720 | One ISP Program Page |
| | ... | One ISP Program Page |
| | 0FE0 | This page includes ROM reserved area. We strongly recommend to reserve the area not to do ISP erase. |

These configurations must be setup completely before starting Page Program. ISP is configured using the following steps:

1. Save program data into IRAM. The data continues for 32 bytes.
2. Set the start address of the content location to PERAM.
3. Set the start address of the anticipated update area to PEROM [15:5]. (By PEROMH/PRROML registers)
4. Write '0xA5A' into PECMD [11:0] to trigger ISP function. Before writing '0x5A' into PECMD[7:0], PECMD[11:8] must be written '0xA'.
5. Write 'NOP' instruction twice.

As an example, assume the 126th page of program memory (IROM, 0x0FC0 – 0x0FDF) is the anticipated update area; the content is already stored in IRAM address 0x60 – 0x7F. To perform the in-system program, simply write starting IROM address 0x0FC0 to EPROMH/EPROML registers, and then specify buffer starting address 0x60 to EPRAM register. Subsequently, write '0xA5A' into PECMD [11:0] registers to duplicate the buffer's data to 126th page of IROM.

In general, every page has the capability to be modified by in-system program procedure. However, since the first and least pages (page 0 and 127) respectively stores reset vector and information for power-on controller, incorrectly perform page program (such as turn off power while programming) may cause faulty power-on sequence / reset.

✷ ***Note:***

***1. Watch dog timer should be clear before the Flash write (program) operation, or watchdog timer would overflow and reset system during ISP operating.***

***2. Don't execute ISP flash ROM program operation for the first page and the last page, or affect program operation.***

## 20.2 In-system Program Register

| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PERAM | PERAM7 | PERAM6 | PERAM5 | PERAM4 | PERAM3 | PERAM2 | PERAM1 | PERAM0 |
| PEROMH | PEROM15 | PEROM14 | PEROM13 | PEROM12 | PEROM11 | PEROM10 | PEROM9 | PEROM8 |
| PEROML | PEROM7 | PEROM6 | PEROM5 | - | PECMD11 | PECMD10 | PECMD9 | PECMD8 |
| PECMD | PECMD7 | PECMD6 | PECMD5 | PECMD4 | PECMD3 | PECMD2 | PECMD1 | PECMD0 |

**PERAM Register (0x97)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | PERAM[7:0] | R/W | 0x00 | The first address of data buffer (IRAM) |

**PEROMH Register (0x96)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | PEROM[15:8] | R/W | 0x00 | The first address ($15^{th}$– $8^{th}$ bit) of program page (IROM) |

**PEROML Register (0x95)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..5 | PEROM[7:5] | R/W | 000 | The first address ($7^{th}$ – $5^{th}$) of program page (IROM) |
| 4 | Reserved | R | 0 | |
| 3..0 | PECMD[11:8] | R/W | 0x0 | 0xA: Enable in-system program<br>Else values: Disable in-system program* |

* Disabling in-system program can avoid mistakenly trigger ISP function.

**PECMD Register (0x94)**

| Bit | Field | Type | Initial | Description |
|---|---|---|---|---|
| 7..0 | PECMD[7:0] | W | 0x0 | 0x5A: Start page program procedure*(1)<br>~~Else values: Reserved~~*(2) |

*(1) Before writing '0x5A' into PECMD[7:0], PECMD[11:8] must be written '0xA'.

*(2) Not permitted to write any other to PECMD register.

## 20.3 Sample Code

```
unsigned cahr idata dataBuffer[32] _at_ 0xE0;  // IRAM 0xE0 to 0xFF

void SYSIspSetDataBuffer(unsigned char address, unsigned char data)
{
  dataBuffer[address & 0x1F] = data;
}

void SYSIspStart(unsigned int pageAddress)
{
  ISP(pageAddress, 0xE0);
}
```

# 21 Electrical Characteristics

## 21.1 Absolute Maximum Ratings

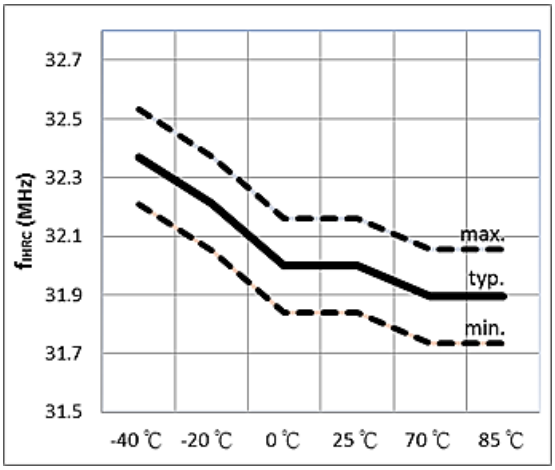
Voltage applied at VDD to VSS ..... - 0.3V to 6.0V
Voltage applied at any pin to VSS..... - 0.3V to VDD+0.3V
Operating ambient temperature .....-40°C to 85°C
Storage ambient temperature .....-40°C to 125°C
Junction Temperature .....-40°C to 125°C

## 21.2 System Operation Characteristics

| | Parameter | Test Condition | Min | TYP | MAX | UNIT |
|---|---|---|---|---|---|---|
| VDD | Operating voltage | fcpu = 1MHz | 1.8 | | 5.5 | V |
| $V_{DR}$ | RAM data retention Voltage | | 1.5 | | | V |
| $V_{POR}$ | VDD rising rate* | | 0.05 | | | V/ms |
| $I_{DD1}$ | Normal mode supply current | VDD = 3V, fcpu = 0.25MHz | | 2.20 | | mA |
| | | VDD = 5V, fcpu = 0.25MHz | | 2.25 | | mA |
| | | VDD = 3V, fcpu = 1MHz | | 2.25 | | mA |
| | | VDD = 5V, fcpu = 1MHz | | 2.30 | | mA |
| | | VDD = 3V, fcpu = 4MHz | | 2.70 | | mA |
| | | VDD = 5V, fcpu = 4MHz | | 2.75 | | mA |
| | | VDD = 3V, fcpu = 8MHz | | 3.20 | | mA |
| | | VDD = 5V, fcpu = 8MHz | | 3.25 | | mA |
| $I_{DD2}$ | STOP mode supply current | VDD = 3V | | 3.5 | 8.5 | µA |
| | | VDD = 5V | | 4.0 | 9.0 | µA |
| $I_{DD3}$ | IDLE mode supply current (fcpu = 1MHz) | VDD = 3V, 32MHz IHRC | | 0.63 | | mA |
| | | VDD = 5V, 32MHz IHRC | | 0.65 | | mA |
| $F_{IHRC}$ | Internal high clock generator | VDD = 1.8V to 5.5V, 25°C | 31.84 | 32 | 32.16 | MHz |
| | | VDD = 1.8V to 5.5V, 25°C to 85°C | 31.68 | - | 31.99 | MHz |
| | | VDD = 1.8V to 5.5V, -40°C to 25°C | 32.31 | - | 32.64 | MHz |
| $F_{ILRC}$ | Internal low clock generator | VDD = 5V, 25°C | 12 | 16 | 24 | kHz |
| $V_{LVD18}$ | LVD18 detect voltage | 25°C | 1.7 | 1.8 | 1.9 | V |
| | | -40°C to 85°C | 1.6 | 1.8 | 2.0 | V |

* Parameter(s) with star mark are non-verified design reference. Ambient temperature is 25°C.

- IHRC Frequency - Temperature Graph




## 21.3 GPIO Characteristics

| | Parameter | Test Condition | MIN | TYP | MAX | UNIT |
|---|---|---|---|---|---|---|
| $V_{IL}$ | Low-level input voltage | | VSS | | 0.3VDD | V |
| $V_{IH}$ | High-level input voltage | | 0.7VDD | | VDD | V |
| $I_{LEKG}$ | I/O port input leakage current | $V_{IN}$ = VDD | | | 2 | µA |
| $R_{UP}$ | Pull-up resister | VDD = 3V | 100 | 200 | 300 | kΩ |
| | | VDD = 5V | 50 | 100 | 150 | kΩ |
| $I_{OH}$ | I/O output source current | VDD = 5V, $V_O$ = VDD-0.5V | 12 | 16 | | mA |
| $I_{OL1}$ | I/O sink current (P07, P1, P2) | VDD = 5V, $V_O$ = VSS+0.5V | 15 | 20 | | mA |
| $I_{OL2}$ | I/O sink current (P00-P06) | VDD = 5V, $V_O$ = VSS+1.5V | 80 | 100 | | mA |

* Ambient temperature is 25°C.

## 21.4 ADC Characteristics

| | Parameter | Test Condition | MIN | TYP | MAX | UNIT |
|---|---|---|---|---|---|---|
| $V_{ADC}$ | Operating voltage | | 2.0 | | 5.5 | V |
| $V_{AIN}$ | AIN channels input voltage | VDD = 5V | 0 | | $V_{REFH}$ | V |
| $V_{REFH}$ | AVREFH pin input voltage | VDD = 5V | 2 | | VDD | V |
| $V_{IREF}$ | Internal VDD reference voltage | VDD = 5V | | VDD | | V |
| | Internal 4V reference voltage | VDD = 5V | 3.92 | 4 | 4.08 | V |
| | Internal 3V reference voltage | VDD = 5V | 2.94 | 3 | 3.06 | V |
| | Internal 2V reference voltage | VDD = 5V | 1.96 | 2 | 2.04 | V |
| $I_{AD}$ | ADC current consumption | VDD = 3V | | 0.65 | | mA |
| | | VDD = 5V | | 0.90 | | mA |
| $f_{ADCLK}$ | ADC clock | VDD = 5V | | | 32 | MHz |
| $f_{ADSMP}$ | ADC sampling rate | VDD = 5V | | | 500 | kHz |
| $t_{ADEN}$ | ADC function enable period | VDD = 5V | 100 | | | µs |
| DNL | Differential nonlinearity* | $f_{ADSMP}$ = 62.5kHz | | ±2 | | LSB |
| | | $f_{ADSMP}$ = 250kHz | | ±2 | | LSB |
| | | $f_{ADSMP}$ = 500kHz | | ±5 | | LSB |
| INL | Integral Nonlinearity* | $f_{ADSMP}$ = 62.5kHz | | ±2 | | LSB |
| | | $f_{ADSMP}$ = 250kHz | | ±2 | | LSB |
| | | $f_{ADSMP}$ = 500kHz | | ±5 | | LSB |
| NMC | No missing code* | $f_{ADSMP}$ = 62.5kHz | 10 | 11 | 12 | Bit |
| | | $f_{ADSMP}$ = 250kHz | | 11 | | Bit |
| | | $f_{ADSMP}$ = 500kHz | | 9 | | Bit |
| $V_{OFFSET}$ | Input offset voltage | Non-trimmed | -10 | 0 | 10 | mV |

* Parameters with star mark: VDD = 5V, $V_{REFH}$ = 2.4V, 25°C.

## 21.5 Flash Memory Characteristics

| | Parameter | Test Condition | MIN | TYP | MAX | UNIT |
|---|---|---|---|---|---|---|
| $V_{dd}$ | Supply voltage | | 1.8 | | 5.5 | V |
| $T_{en}$ | Endurance time | 25°C | | *100K | | cycle |
| $I_{wrt}$ | Write current | 25°C | | 3 | 4 | mA |
| $T_{wrt}$ | Write time | Write 1 page=32 bytes, 25°C | | 6 | 8 | ms |

* Parameters with star mark are non-verified design reference.

# 22 Instruction Set

This chapter categorizes the SN8F5702 microcontroller's comprehensive assembly instructions. It includes five categories—arithmetic operation, logic operation, data transfer operation, Boolean manipulation, and program branch—which are fully compatible with standard 8051.

**Symbol description**

| Symbol | Description |
|---|---|
| Rn | Working register R0 - R7 |
| direct | One of 128 internal RAM locations or any Special Function Register |
| @Ri | Indirect internal or external RAM location addressed by register R0 or R1 |
| #data | 8-bit constant (immediate operand) |
| #data16 | 16-bit constant (immediate operand) |
| bit | One of 128 software flags located in internal RAM, or any flag of bit-addressable Special Function Registers |
| addr16 | Destination address for LCALL or LJMP, can be anywhere within the 64-Kbyte page of program memory address space |
| addr11 | Destination address for ACALL or AJMP, within the same 2-Kbyte page of program memory as the first byte of the following instruction |
| rel | SJMP and all conditional jumps include an 8-bit offset byte. Its range is +127/-128 bytes relative to the first byte of the following instruction |
| A | Accumulator |

**Arithmetic operations**

| Mnemonic | Description |
|---|---|
| ADD A, Rn | Add register to accumulator |
| ADD A, direct | Add directly addressed data to accumulator |
| ADD A, @Ri | Add indirectly addressed data to accumulator |
| ADD A, #data | Add immediate data to accumulator |
| ADDC A, Rn | Add register to accumulator with carry |
| ADDC A, direct | Add directly addressed data to accumulator with carry |
| ADDC A, @Ri | Add indirectly addressed data to accumulator with carry |
| ADDC A, #data | Add immediate data to accumulator with carry |
| SUBB A, Rn | Subtract register from accumulator with borrow |
| SUBB A, direct | Subtract directly addressed data from accumulator with borrow |
| SUBB A, @Ri | Subtract indirectly addressed data from accumulator with borrow |
| SUBB A, #data | Subtract immediate data from accumulator with borrow |
| INC A | Increment accumulator |
| INC Rn | Increment register |
| INC direct | Increment directly addressed location |
| INC @Ri | Increment indirectly addressed location |
| INC DPTR | Increment data pointer |
| DEC A | Decrement accumulator |
| DEC Rn | Decrement register |
| DEC direct | Decrement directly addressed location |
| DEC @Ri | Decrement indirectly addressed location |
| MUL AB | Multiply A and B |
| DIV | Divide A by B |
| DA A | Decimally adjust accumulator |

**Logic operations**

| Mnemonic | Description |
|---|---|
| ANL A, Rn | AND register to accumulator |
| ANL A, direct | AND directly addressed data to accumulator |
| ANL A, @Ri | AND indirectly addressed data to accumulator |
| ANL A, #data | AND immediate data to accumulator |
| ANL direct, A | AND accumulator to directly addressed location |
| ANL direct, #data | AND immediate data to directly addressed location |
| ORL A, Rn | OR register to accumulator |

| | |
|---|---|
| ORL A, direct | OR directly addressed data to accumulator |
| ORL A, @Ri | OR indirectly addressed data to accumulator |
| ORL A, #data | OR immediate data to accumulator |
| ORL direct, A | OR accumulator to directly addressed location |
| ORL direct, #data | OR immediate data to directly addressed location |
| XRL A, Rn | Exclusive OR (XOR) register to accumulator |
| XRL A, direct | XOR directly addressed data to accumulator |
| XRL A, @Ri | XOR indirectly addressed data to accumulator |
| XRL A, #data | XOR immediate data to accumulator |
| XRL direct, A | XOR accumulator to directly addressed location |
| XRL direct, #data | XOR immediate data to directly addressed location |
| CLR A | Clear accumulator |
| CPL A | Complement accumulator |
| RL A | Rotate accumulator left |
| RLC A | Rotate accumulator left through carry |
| RR A | Rotate accumulator right |
| RRC A | Rotate accumulator right through carry |
| SWAP A | Swap nibbles within the accumulator |

**Data transfer operations**

| Mnemonic | Description |
|---|---|
| MOV A, Rn | Move register to accumulator |
| MOV A, direct | Move directly addressed data to accumulator |
| MOV A, @Ri | Move indirectly addressed data to accumulator |
| MOV A, #data | Move immediate data to accumulator |
| MOV Rn, A | Move accumulator to register |
| MOV Rn, direct | Move directly addressed data to register |
| MOV Rn, #data | Move immediate data to register |
| MOV direct, A | Move accumulator to direct |
| MOV direct, Rn | Move register to direct |
| MOV direct1, direct2 | Move directly addressed data to directly addressed location |
| MOV direct, @Ri | Move indirectly addressed data to directly addressed location |
| MOV direct, #data | Move immediate data to directly addressed location |
| MOV @Ri, A | Move accumulator to indirectly addressed location |
| MOV @Ri, direct | Move directly addressed data to indirectly addressed location |
| MOV @Ri, #data | Move immediate data to in directly addressed location |

| | |
|---|---|
| MOV DPTR, #data16 | Load data pointer with a 16-bit immediate |
| MOVC A, @A+DPTR | Load accumulator with a code byte relative to DPTR |
| MOVC A, @A+PC | Load accumulator with a code byte relative to PC |
| MOVX A, @Ri | Move external RAM (8-bit address) to accumulator |
| MOVX A, @DPTR | Move external RAM (16-bit address) to accumulator |
| MOVX @Ri, A | Move accumulator to external RAM (8-bit address) |
| MOVX @DPTR, A | Move accumulator to external RAM (16-bit address) |
| PUSH direct | Push directly addressed data onto stack |
| POP direct | Pop directly addressed location from stack |
| XCH A, Rn | Exchange register with accumulator |
| XCH A, direct | Exchange directly addressed location with accumulator |
| XCH A, @Ri | Exchange indirect RAM with accumulator |
| XCHD A, @Ri | Exchange low-order nibbles of indirect and accumulator |

**Boolean manipulation**

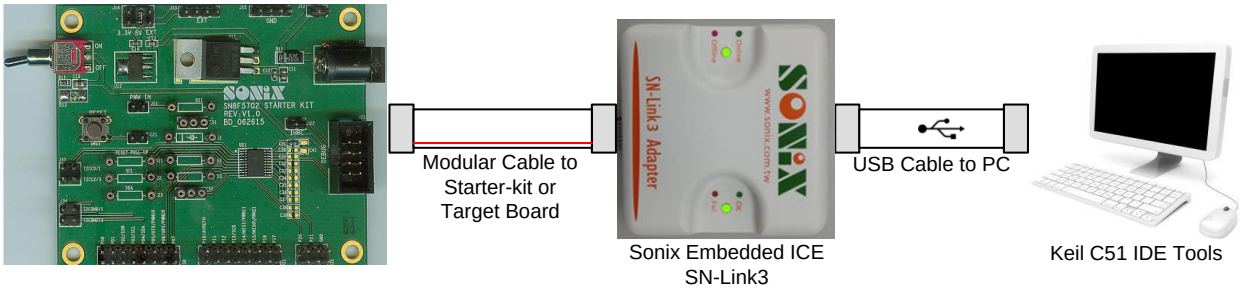
| Mnemonic | Description |
|---|---|
| CLR C | Clear carry flag |
| CLR bit | Clear directly addressed bit |
| SETB C | Set carry flag |
| SETB bit | Set directly addressed bit |
| CPL C | Complement carry flag |
| CPL bit | Complement directly addressed bit |
| ANL C, bit | AND directly addressed bit to carry flag |
| ANL C, /bit | AND complement of directly addressed bit to carry |
| ORL C, bit | OR directly addressed bit to carry flag |
| ORL C, /bit | OR complement of directly addressed bit to carry |
| MOV C, bit | Move directly addressed bit to carry flag |
| MOV bit, C | Move carry flag to directly addressed bit |

**Program branches**

| Mnemonic | Description |
|---|---|
| ACALL addr11 | Absolute subroutine call |
| LCALL addr16 | Long subroutine call |
| RET | Return from subroutine |
| RETI | Return from interrupt |
| AJMP addr11 | Absolute jump |
| LJMP addr16 | Long jump |
| SJMP rel | Short jump (relative address) |
| JMP @A+DPTR | Jump indirect relative to the DPTR |
| JZ rel | Jump if accumulator is zero |
| JNZ rel | Jump if accumulator is not zero |
| JC rel | Jump if carry flag is set |
| JNC rel | Jump if carry flag is not set |
| JB bit, rel | Jump if directly addressed bit is set |
| JNB bit, rel | Jump if directly addressed bit is not set |
| JBC bit, rel | Jump if directly addressed bit is set and clear bit |
| CJNE A, direct, rel | Compare directly addressed data to accumulator and jump if not equal |
| CJNE A, #data, rel | Compare immediate data to accumulator and jump if not equal |
| CJNE Rn, #data, rel | Compare immediate data to register and jump if not equal |
| CJNE @Ri, #data, rel | Compare immediate to indirect and jump if not equal |
| DJNZ Rn, rel | Decrement register and jump if not zero |
| DJNZ direct, rel | Decrement directly addressed location and jump if not zero |
| NOP | No operation for one cycle |

# 23 Development Environment

SONIX provides an Embedded ICE emulator system to offer SN8F5702 firmware development. The platform is an in-circuit debugger and controlled by Keil C51 IDE software on Microsoft Windows platform. The platform includes SN-Link3, SN8F5702 Starter-kit and Keil C51 IDE software to build a high-speed, low cost, powerful and multi-task development environment including emulator, debugger and programmer. To execute emulation is like run real chip because the emulator circuit integrated in SN8F5702 to offer a real development environment.




## 23.1 Minimum Requirement

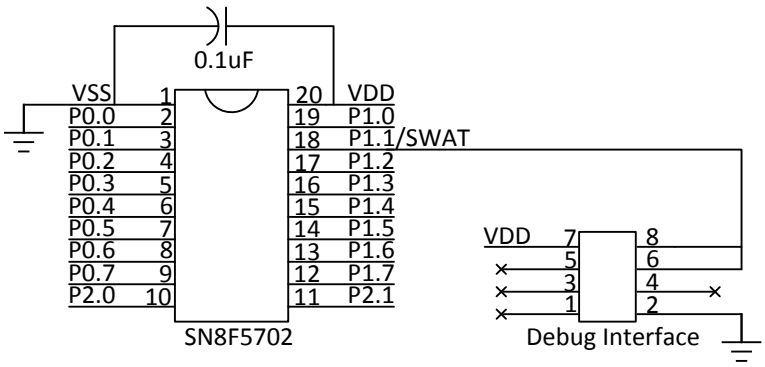
The following items are essential to build up an appropriate development environment. The compatibility is verified on listed versions, and is expected to execute perfectly on later version. SN-Link related information is available to download on SONiX website (www.sonix.com.tw);
Keil C51 is downloadable on www.keil.com/c51.

- **SN-Link3 Adapter** with updated firmware version 1.02
- **SN-Link Driver for Keil C51** version 1.00.317
- **Keil C51** version 9.50a and 9.54a or greater.

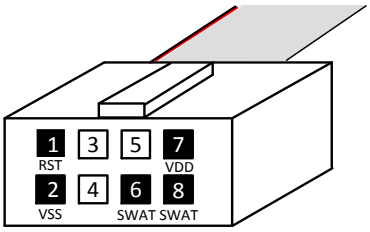
## 23.2 Debug Interface Hardware

The circuit below demonstrates the appropriate method to connect microcontroller's SWAT pin and SN-Link3 Adapter.

Before starting debug, microcontroller's power (VDD) must be switched off. Connect the SWAT to both 6$^{th}$ and 8$^{th}$ pins of SN-Link, and respectively link VDD and VSS to 7$^{th}$ pin and 2$^{nd}$ pin. A handshake procedure would be automatically started by turn on the microcontroller, and SN-Link's green LED (Run) indicates the success of connection (refer *SN8F5000 Debug Tool Manual* for further detail).




example circuit

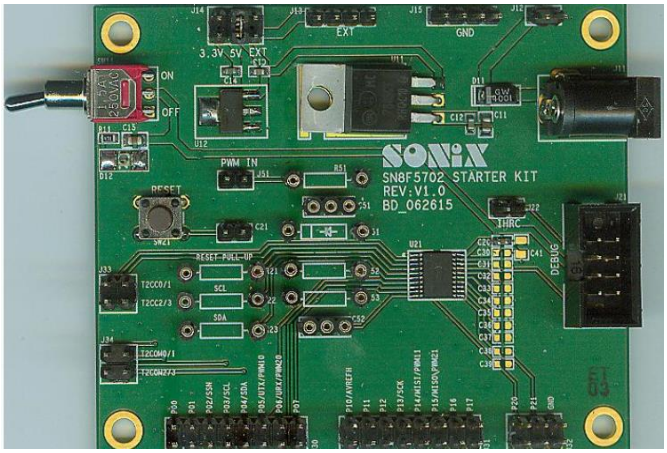



SN-Link header

## 23.3 Development Tool

SN-Link3 Adapter


Starter-Kit support SN8F5702, SN8F570200/202, SN8F570210/211/212/213




MP5 Writer

# 24 SN8F5702 Starter-Kit

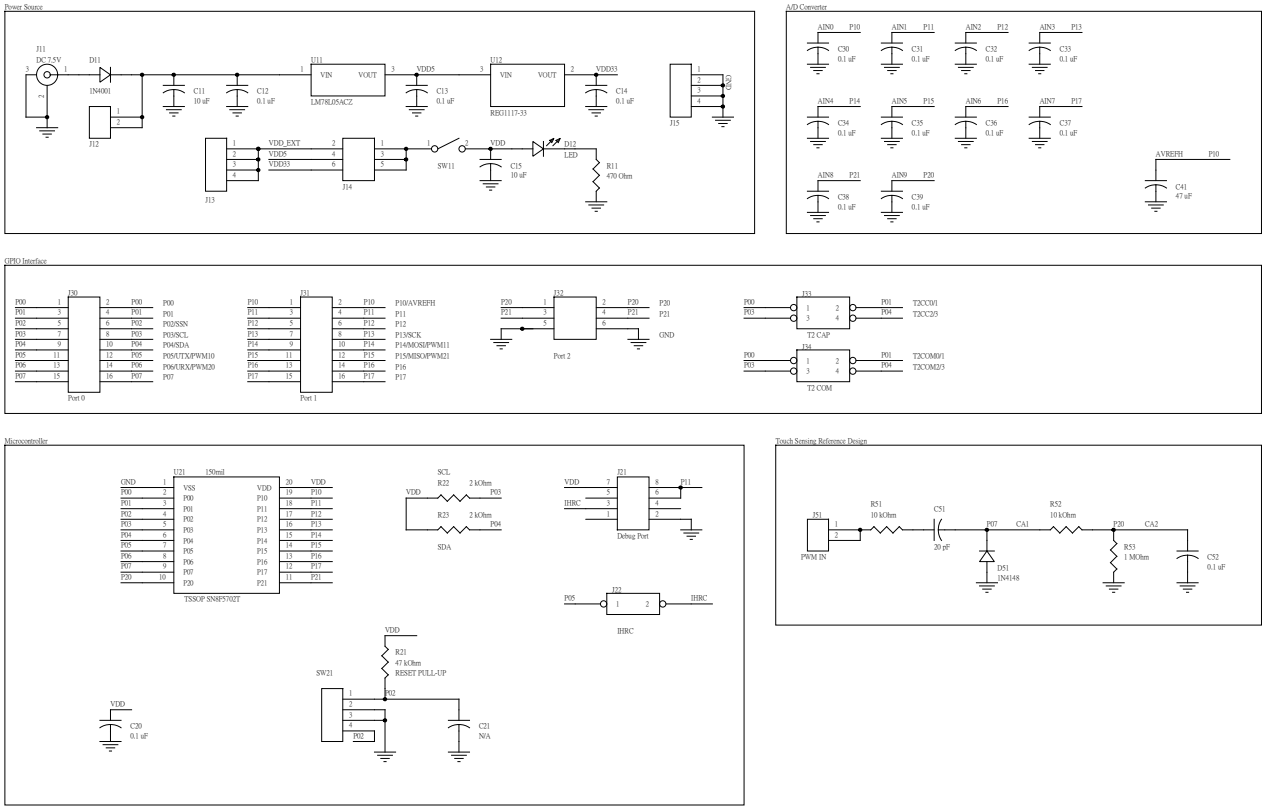
SN8F5000 Starter-Kit provides easy-development platform. It includes SN8F5000 family real chip and I/O connectors to input signal or drive device of user's application. It is a simple platform to develop application as target board not ready. The Starter-Kit can be replaced by target board, because SN8F5000 family integrates embedded ICE in-circuit debugger circuitry.

## 24.1 Configurations of Circuit

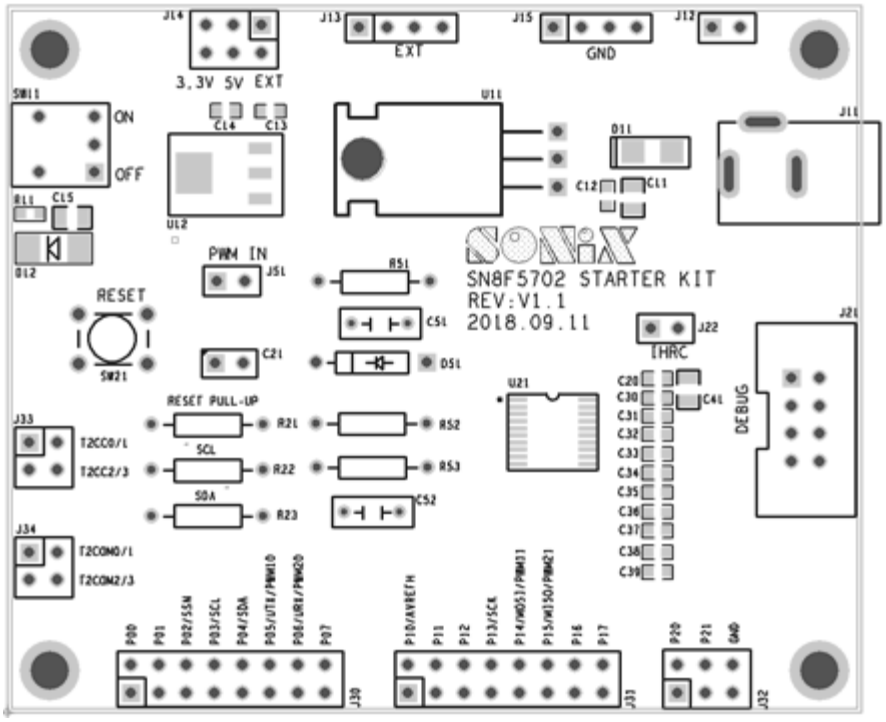
These configurations must be setup completely before starting Starter-Kit developing.

1. Confirm to the circuit board whether elements are complete.
2. The power source of Starter-Kit circuit is chosen from 5.0V, 3.3V, external power or Micro USB via jumper.
3. The power source comes from 5.0V or 3.3V which must be connect to DC 7.5V power adapter.
4. If the power source is chosen from external power, then external power source connects to EXT pin.
5. The "RST" pin needs to connect pull high resister to VDD when external reset is chosen to use.
6. The Debug Port can connect SN-LINK Adapter for emulation or download code.
7. The MCU LED will light up and SN8F5000 family chip will be connected to power when power (VDD) is switched on.

## 24.2 Schematic




## 24.3 Floor Plan of PCB layout



## 24.4 Component Description

| Number | Description |
|---|---|
| C30 – C39 | 10-ch ADC capacitors. |
| C41 | AVREFH capacitor. |
| D12 | MCU LED |
| J11 | DC 7.5V power adapter |
| J13/J15 | External power source. |
| SW21 | External reset trigger source |
| J14 | VDD power source is 5.0V, 3.3V or external power. |
| J21 | Debug Port |
| J30 – J32 | I/O connector. |
| J33 | Timer 2 capture connector. |
| J34 | Timer 2 compare connector. |
| R21, C21 | External reset pull-high resister and capacitor. |
| R22, R23 | I2C pull-high resisters. |
| SW11 | Target power (VDD) switch |
| U21 | SN8F5702T real chip (Sonix standard option). |

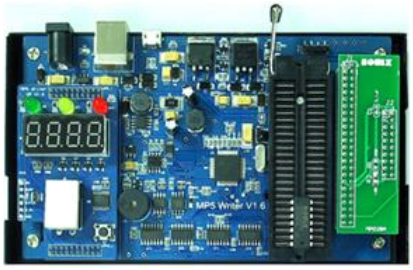
# 25 ROM Programming Pin

SN8F5702 Series Flash ROM erase/program/verify support SN-Link and MP5 Writer

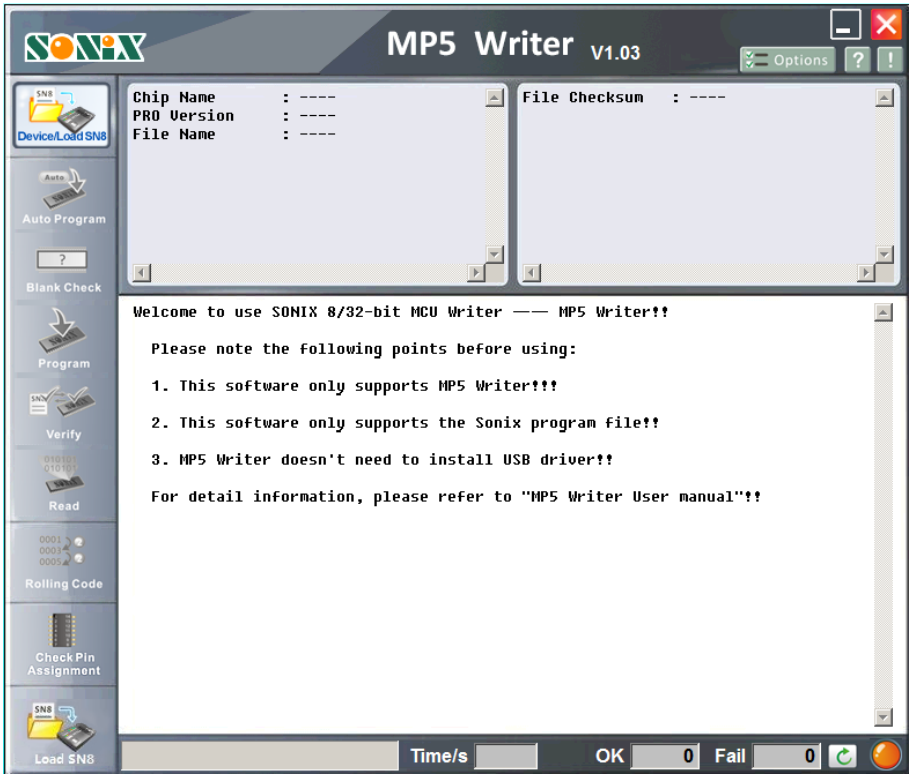
- SN-Link: Debug interface and on board programming.
- MP5 Writer: For SN8F5702 series version mass programming.

## 25.1 MP5 Hardware Connecting

Different package type with MCU programming connecting is as following, DIP, SOP, SSOP, TSSOP and QFN Illustration.

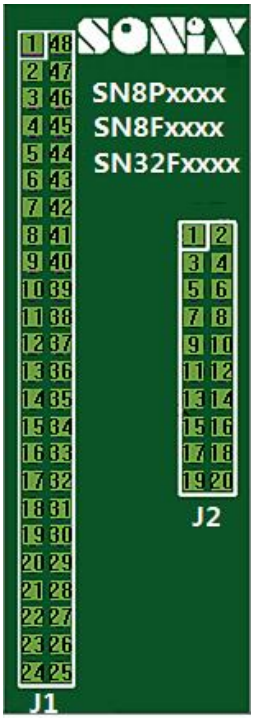


MP5 Software operation interface is as following.



## 25.2 MP5 Writer Transition Board Socket Pin Assignment

MP5 Writer Transition Board:




## 25.3 MP5 Writer Programming Pin Mapping

| Writer Connector | | MCU Pin Number | SN8F5702P/S/T | | SN8F5702J | | SN8F570200A | | SN8F570202S | |
|---|---|---|---|---|---|---|---|---|---|---|
| J2 Pin Number | J2 Pin Name | | MCU Pin Number | J1 Pin Number | MCU Pin Number | J1 Pin Number | MCU Pin Number | J1 Pin Number | MCU Pin Number | J1 Pin Number |
| 1 | VDD | VDD | 20 | 34 | 18 | 32 | 1 | 20 | 1 | 21 |
| 2 | GND | VSS | 1 | 15 | 19 | 33 | 2 | 21 | 8 | 28 |
| 7 | SWAT | P1.1 | 18 | 32 | 16 | 30 | 9 | 28 | 5 | 25 |
| 9 | SWAT | P1.1 | 18 | 32 | 16 | 30 | 9 | 28 | 5 | 25 |
| 20 | PDB | P0.5 | 7 | 21 | 5 | 19 | 5 | 24 | 3 | 23 |

| Writer Connector | | MCU Pin Number | SN8F570210S | | SN8F570211S | | SN8F570212S/T | | SN8F570213S | |
|---|---|---|---|---|---|---|---|---|---|---|
| J2 Pin Number | J2 Pin Name | | MCU Pin Number | J1 Pin Number | MCU Pin Number | J1 Pin Number | MCU Pin Number | J1 Pin Number | MCU Pin Number | J1 Pin Number |
| 1 | VDD | VDD | 14 | 31 | 14 | 31 | 16 | 32 | 1 | 18 |
| 2 | GND | VSS | 1 | 18 | 1 | 18 | 1 | 17 | 14 | 31 |
| 7 | SWAT | P1.1 | 12 | 29 | 12 | 29 | 14 | 30 | 12 | 29 |
| 9 | SWAT | P1.1 | 12 | 29 | 12 | 29 | 14 | 30 | 12 | 29 |
| 20 | PDB | P0.5 | 5 | 22 | 6 | 23 | 6 | 22 | 5 | 22 |

## 25.4 SN-Link ISP Programming

SN-Link ISP programming hardware and software are as following.



## 25.5 SN-Link ISP Programming Pin Mapping

| SN-Link Connector | | MCU Pin Number | SN8F5702P/S/T | SN8F5702J | SN8F570200A | SN8F570202S |
|---|---|---|---|---|---|---|
| Pin Number | Pin Name | | Pin Number | Pin Number | Pin Number | Pin Number |
| 7 | VDD | VDD | 20 | 18 | 1 | 1 |
| 2 | GND | VSS | 1 | 19 | 2 | 8 |
| 6 | SWAT | P1.1 | 18 | 16 | 9 | 5 |
| 8 | SWAT | P1.1 | 18 | 16 | 9 | 5 |

| SN-Link Connector | | MCU Pin Number | SN8F570210S | SN8F570211S | SN8F570212S/T | SN8F570213S |
|---|---|---|---|---|---|---|
| Pin Number | Pin Name | | Pin Number | Pin Number | Pin Number | Pin Number |
| 7 | VDD | VDD | 14 | 14 | 16 | 1 |
| 2 | GND | VSS | 1 | 1 | 1 | 14 |
| 6 | SWAT | P1.1 | 12 | 12 | 14 | 12 |
| 8 | SWAT | P1.1 | 12 | 12 | 14 | 12 |

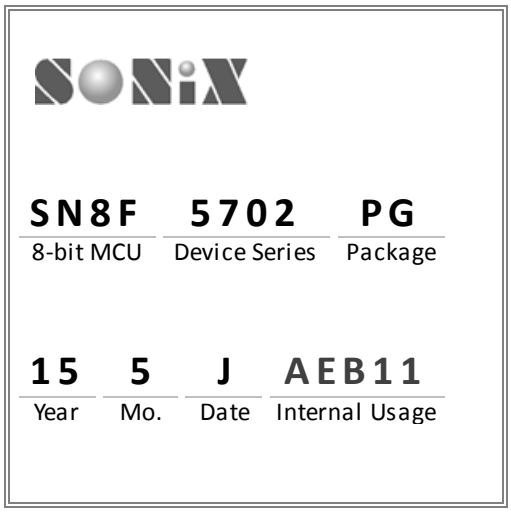
# 26 Ordering Information

A typical surface of SONiX microcontroller is printed with three columns: logo, device's full name, and date code.

SONiX Logo

Full Name

Date Code




## 26.1 Device Nomenclature

| Full Name | Packing Type |
|---|---|
| S8F5702W | Wafer |
| SN8F5702H | Dice |
| SN8F5702PG | PDIP, 20 pins, Green package |
| SN8F5702SG | SOP, 20 pins, Green package |
| SN8F5702TG | TSSOP, 20 pins, Green package |
| SN8F5702JG | QFN, 20 pins, Green package |
| SN8F570200AG | MSOP, 10 pins, Green package |
| SN8F570202SG | SOP, 8 pins, Green package |
| SN8F570210SG | SOP, 14 pins, Green package |
| SN8F570211SG | SOP, 14 pins, Green package |
| SN8F570212SG | SOP, 16 pins, Green package |
| SN8F570212TG | TSSOP, 16 pins, Green package |
| SN8F570213SG | SOP, 14 pins, Green package |

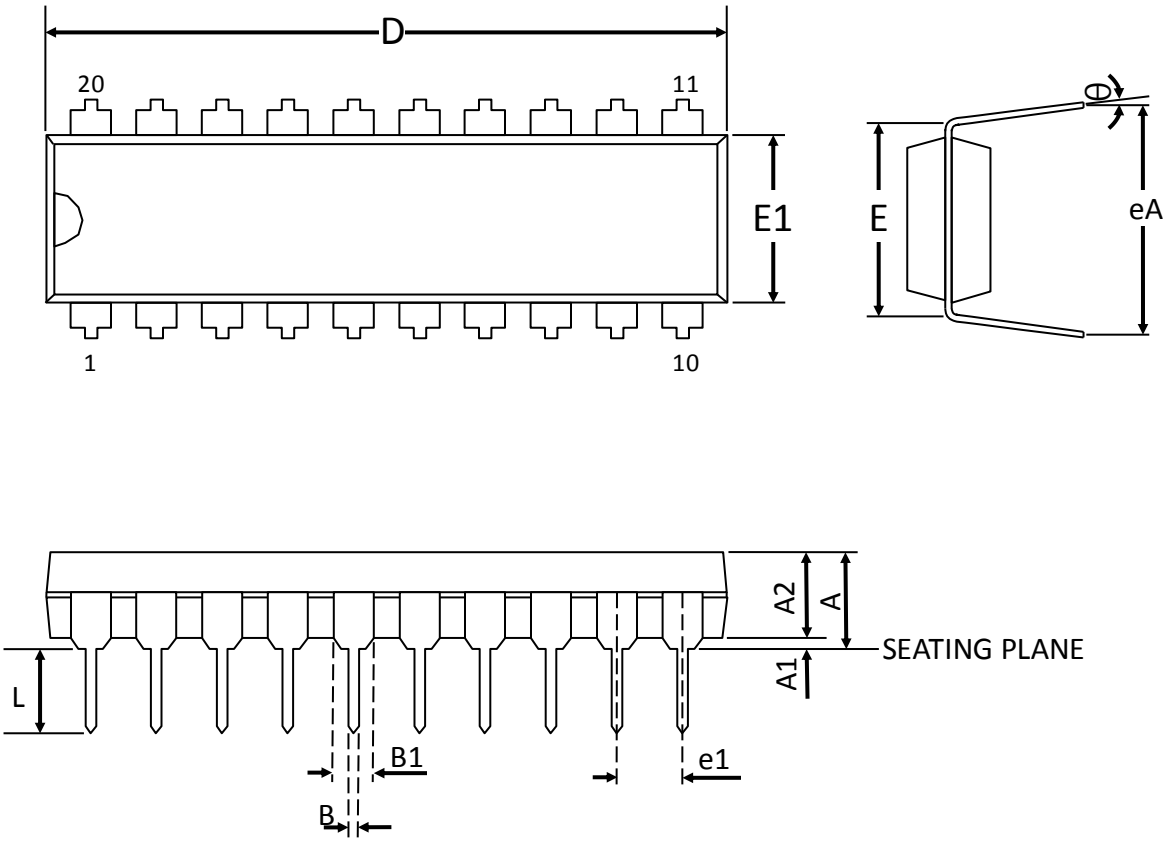
## 26.2 Date Code

The date code includes two parts: date of manufacture and production serial code. The first part is public information which is encoded by following principles.

| | |
|---|---|
| Year | 15: 2015<br>16: 2016<br>17: 2017<br>et cetera |
| Month | 1: January<br>2: February<br>3: March<br>A: October<br>B: November<br>C: December<br>et cetera |
| Date | 1: 01<br>2: 02<br>3: 03<br>A: 10<br>B: 11<br>et cetera |

# 27 Package Information

## 27.1 DIP20

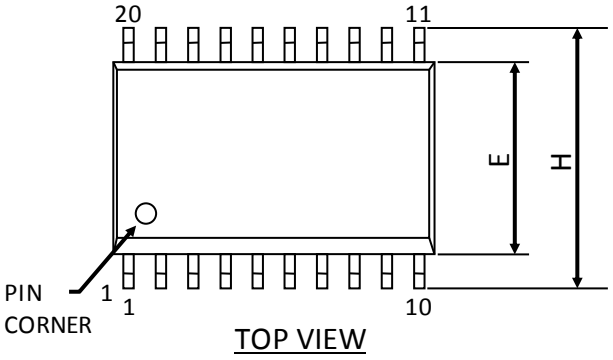



| SYMBOLS | Dimension in mm | | | Dimension in inch | | |
|---|---|---|---|---|---|---|
| | MIN. | NOM. | MAX. | MIN. | NOM. | MAX. |
| A | -- | -- | 4.45 | -- | -- | 0.175 |
| A1 | 0.35 | -- | -- | 0.015 | -- | -- |
| A2 | 3.18 | 3.30 | 3.43 | 0.125 | 0.130 | 0.135 |
| B | 0.46 typ. | | | 0.018 typ. | | |
| B1 | 1.52 typ. | | | 0.060 typ. | | |
| D | 25.70 | 26.06 | 26.42 | 1.012 | 1.026 | 1.040 |
| E | 7.62 BSC. | | | 0.300 BSC. | | |
| E1 | 6.05 | 6.35 | 6.65 | 0.238 | 0.250 | 0.261 |
| e1 | 2.54 typ. | | | 0.100 typ. | | |
| L | 3.05 | 3.30 | 3.56 | 0.120 | 0.130 | 0.140 |
| eA | 7.62 | 9.02 | 9.53 | 0.300 | 0.355 | 0.375 |
| θ | 0° | 7° | 15° | 0° | 7° | 15° |

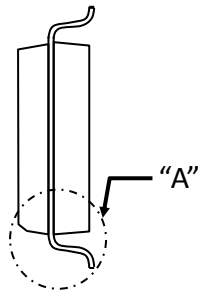
Notes：

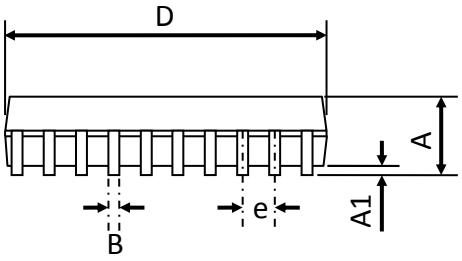
1. JEDEC OUTLINE：MS-001 AD
2. CONTROLLING DIMENSION：inch

## 27.2 SOP20

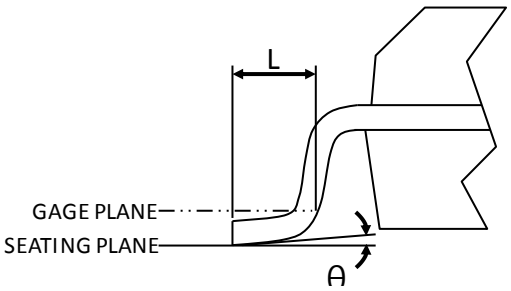



TOP VIEW




SIDE VIEW



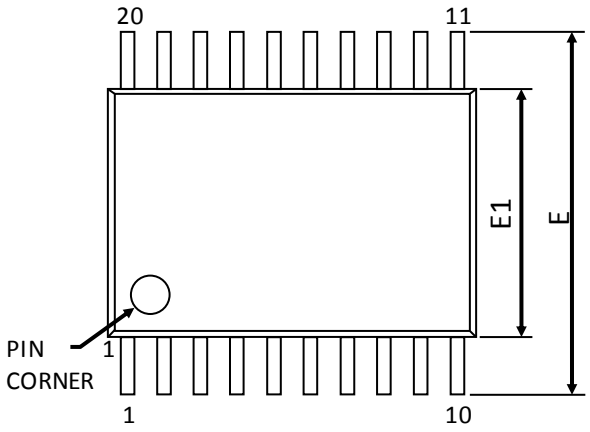

DETAIL "A"

| SYMBOLS | Dimension in mm | | | Dimension in inch | | |
|---|---|---|---|---|---|---|
| | MIN. | NOM. | MAX. | MIN. | NOM. | MAX. |
| A | -- | -- | 2.65 | -- | -- | 0.104 |
| A1 | 0.10 | -- | 0.30 | 0.004 | -- | 0.012 |
| B | 0.31 | 0.41 | 0.51 | 0.012 | 0.016 | 0.020 |
| D | 12.80 BSC | | | 0.503 | | |
| E | 7.50 BSC | | | 0.295 | | |
| e | 1.27 BSC | | | 0.050 BSC | | |
| H | 10.30 BSC | | | 0.405 | | |
| L | 0.40 | -- | 1.27 | 0.016 | -- | 0.050 |
| θ | 0° | 4° | 8° | 0° | 4° | 8° |

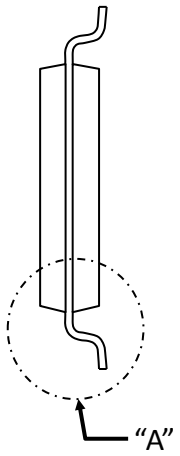
Notes：

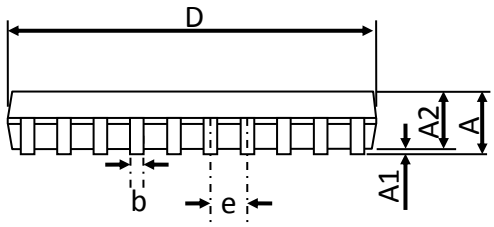
1. CONTROLLING DIMENSION：mm
2. JEDEC OUTLINE：MO-013 AC

## 27.3 TSSOP20

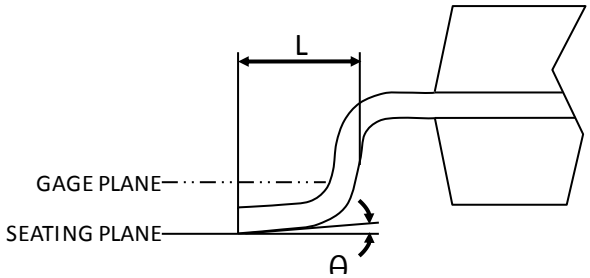



TOP VIEW




SIDE VIEW



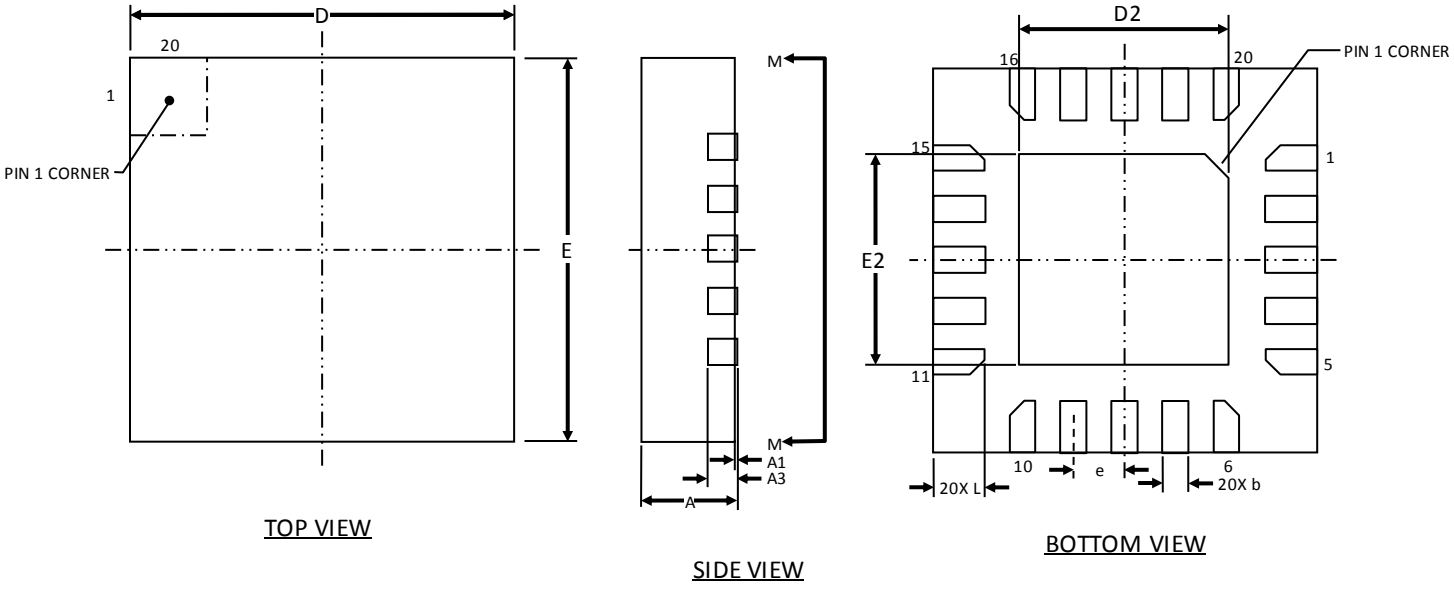

DETAIL "A"

| SYMBOLS | Dimension in mm | | | Dimension in inch | | |
|---|---|---|---|---|---|---|
| | MIN. | NOM. | MAX. | MIN. | NOM. | MAX. |
| A | -- | -- | 1.20 | -- | -- | 0.047 |
| A1 | 0.05 | -- | 0.15 | 0.002 | -- | 0.006 |
| A2 | 0.80 | -- | 1.05 | 0.031 | -- | 0.041 |
| b | 0.19 | -- | 0.30 | 0.007 | -- | 0.012 |
| D | 6.40 | 6.50 | 6.60 | 0.252 | 0.256 | 0.260 |
| E | | 6.40 BSC. | | | 0.252 BSC. | |
| E1 | 4.30 | 4.40 | 4.50 | 0.169 | 0.173 | 0.177 |
| e | | 0.65 BSC. | | | 0.026 BSC. | |
| L | 0.45 | 0.60 | 0.75 | 0.018 | 0.024 | 0.030 |
| θ | 0° | -- | 8° | 0° | -- | 8° |

Notes：

1. CONTROLLING DIMENSION：mm
2. JEDEC OUTLINE：MO-153
3. DIMENSION 'D' DOES NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BERRES.
4. DIMENSION 'E1' DOES NOT INCLUDE INTERLEAD FLASH OR PROTRUSION.
5. DIMENSION 'b' DOES NOT INCLUDE DAMBAR PROTRUSION.

## 27.4 QFN20 3X3




TOP VIEW

SIDE VIEW

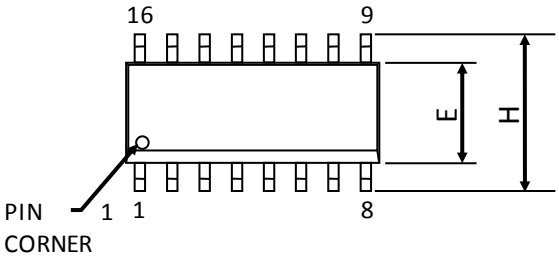
BOTTOM VIEW

| SYMBOLS | Dimension in mm | | | Dimension in inch | | |
|---|---|---|---|---|---|---|
| | MIN. | NOM. | MAX. | MIN. | NOM. | MAX. |
| A | 0.70 | 0.80 | 0.90 | 0.028 | 0.031 | 0.035 |
| A1 | 0.00 | 0.02 | 0.05 | 0.000 | 0.001 | 0.002 |
| A3 | 0.203 REF | | | 0.008 REF | | |
| b | 0.15 | 0.20 | 0.25 | 0.006 | 0.008 | 0.010 |
| D | 3.00 BSC | | | 0.118 BSC | | |
| E | 3.00 BSC | | | 0.118 BSC | | |
| e | 0.40 BSC | | | 0.016 BSC | | |
| D2 | 1.55 | 1.65 | 1.75 | 0.61 | 0.65 | 0.69 |
| E2 | 1.55 | 1.65 | 1.75 | 0.61 | 0.65 | 0.69 |
| L | 0.30 | 0.40 | 0.50 | 0.012 | 0.016 | 0.020 |

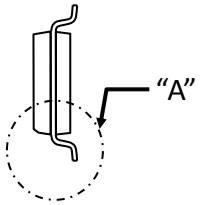
Notes：

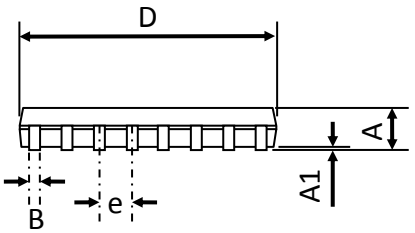
1. CONTROLLING DIMENSION：MILLIMETER (mm)

## 27.5 SOP16

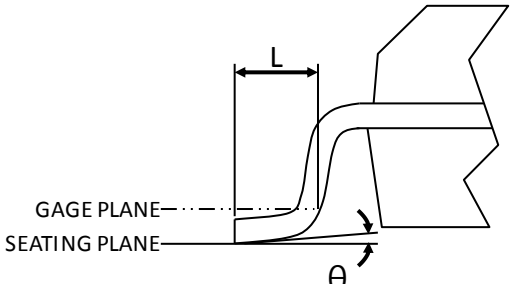



TOP VIEW




SIDE VIEW



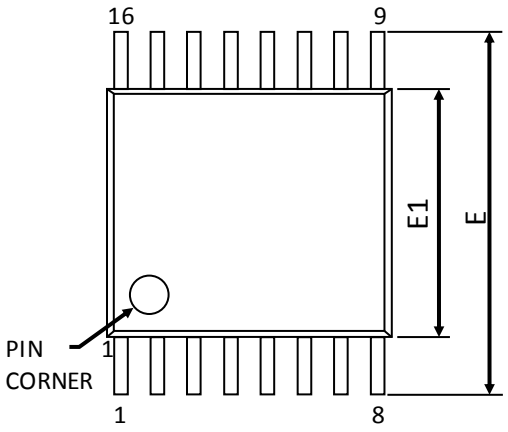

DETAIL "A"

| SYMBOLS | Dimension in mm | | | Dimension in inch | | |
|---|---|---|---|---|---|---|
| | MIN. | NOM. | MAX. | MIN. | NOM. | MAX. |
| A | -- | -- | 1.75 | -- | -- | 0.069 |
| A1 | 0.10 | -- | 0.25 | 0.004 | -- | 0.010 |
| B | 0.31 | 0.41 | 0.51 | 0.012 | 0.016 | 0.020 |
| D | 9.90 BSC | | | 0.389 BSC | | |
| E | 3.90 BSC | | | 0.153 BSC | | |
| e | 1.27 BSC | | | 0.050 BSC | | |
| H | 6.00 BSC | | | 0.236 BSC | | |
| L | 0.40 | -- | 1.27 | 0.016 | -- | 0.050 |
| θ | 0° | 4° | 8° | 0° | 4° | 8° |

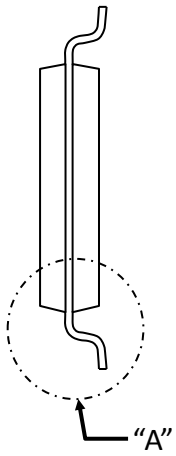
Notes：

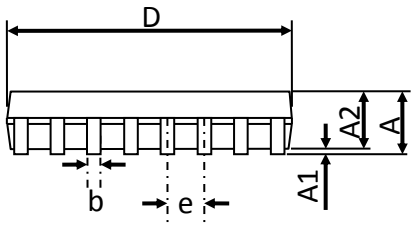
1. CONTROLLING DIMENSION：mm
2. JEDEC OUTLINE：MS-012 AC

## 27.6 TSSOP16

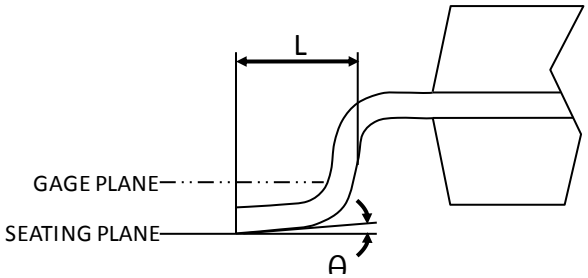



TOP VIEW




SIDE VIEW



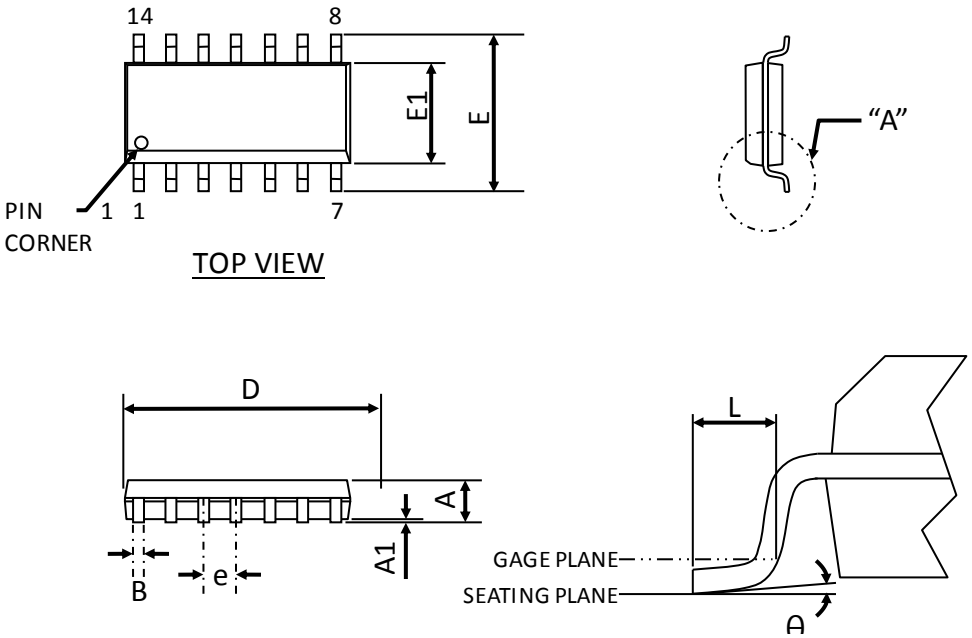

DETAIL "A"

| SYMBOLS | Dimension in mm | | | Dimension in inch | | |
|---|---|---|---|---|---|---|
| | MIN. | NOM. | MAX. | MIN. | NOM. | MAX. |
| A | -- | -- | 1.20 | -- | -- | 0.047 |
| A1 | 0.05 | -- | 0.15 | 0.002 | -- | 0.006 |
| A2 | 0.80 | 1.00 | 1.05 | 0.031 | 0.039 | 0.041 |
| b | 0.19 | -- | 0.30 | 0.007 | -- | 0.012 |
| D | 4.80 | 5.00 | 5.20 | 0.189 | 0.196 | 0.205 |
| E | 6.40 BSC. | | | 0.252 BSC. | | |
| E1 | 4.30 | 4.40 | 4.50 | 0.169 | 0.173 | 0.177 |
| e | 0.65 BSC. | | | 0.026 BSC. | | |
| L | 0.45 | 0.60 | 0.75 | 0.018 | 0.024 | 0.030 |
| θ | 0° | -- | 8° | 0° | -- | 8° |

Notes：

1. CONTROLLING DIMENSION：mm
2. JEDEC OUTLINE：MO-153
3. DIMENSION 'D' DOES NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BERRES.
4. DIMENSION 'E1' DOES NOT INCLUDE INTERLEAD FLASH OR PROTRUSION.
5. DIMENSION 'b' DOES NOT INCLUDE DAMBAR PROTRUSION.

## 27.7 SOP14

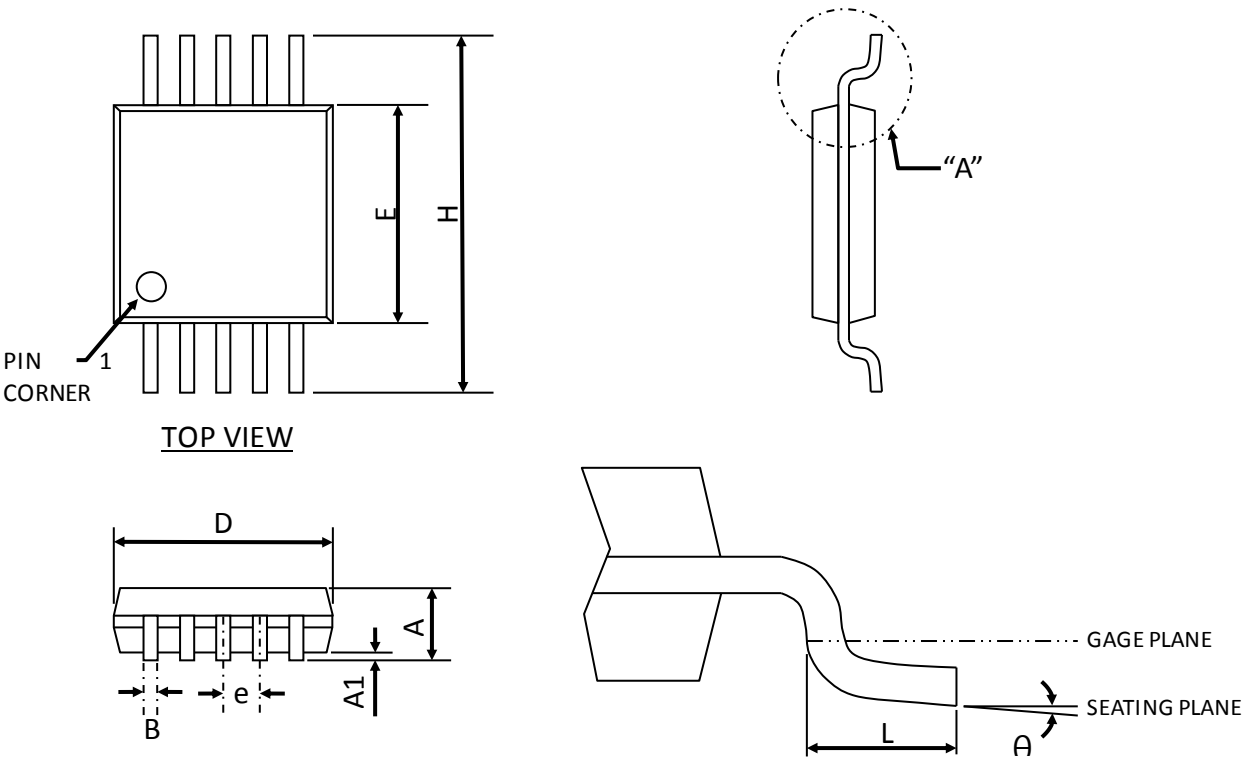



| SYMBOLS | Dimension in mm | | | Dimension in inch | | |
|---|---|---|---|---|---|---|
| | MIN. | NOM. | MAX. | MIN. | NOM. | MAX. |
| A | -- | -- | 1.75 | -- | -- | 0.069 |
| A1 | 0.05 | -- | 0.25 | 0.002 | -- | 0.010 |
| B | 0.31 | -- | 0.51 | 0.012 | -- | 0.02 |
| D | 8.65 BSC | | | 0.340 BSC | | |
| E1 | 3.90 BSC | | | 0.154 BSC | | |
| e | 1.27 BSC | | | 0.050 BSC | | |
| E | 6.00 BSC | | | 0.236 BSC | | |
| L | 0.4 | -- | 1.27 | 0.015 | -- | 0.050 |
| θ | 0° | -- | 8° | 0° | -- | 8° |

Notes：

1. CONTROLLING DIMENSION：mm
2. JEDEC OUTLINE：MS-012 AB

## 27.8 MSOP10

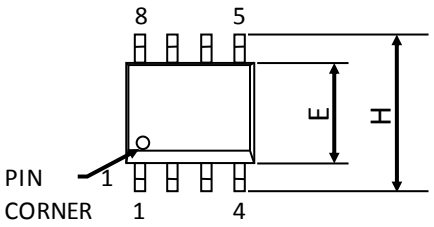



| SYMBOLS | Dimension in mm | | | Dimension in inch | | |
|---|---|---|---|---|---|---|
| | MIN. | NOM. | MAX. | MIN. | NOM. | MAX. |
| A | -- | -- | 1.10 | -- | -- | 0.043 |
| A1 | 0.00 | -- | 0.15 | 0.000 | -- | 0.006 |
| B | 0.17 | -- | 0.27 | 0.007 | -- | 0.011 |
| D | 3.00 BSC | | | 0.118 BSC | | |
| E | 3.00 BSC | | | 0.118 BSC | | |
| e | 0.50 BSC | | | 0.020 BSC | | |
| H | 4.9 BSC | | | 0.193 BSC | | |
| L | 0.40 | 0.60 | 0.80 | 0.016 | 0.024 | 0.031 |
| θ | 0° | 4° | 8° | 0° | 4° | 8° |

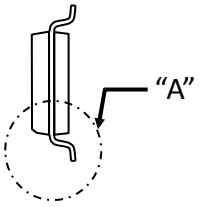
Notes：

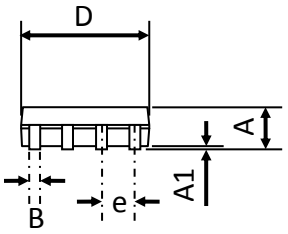
1. CONTROLLING DIMENSION：mm
2. JEDEC OUTLINE：MO-187 BA

## 27.9 SOP8

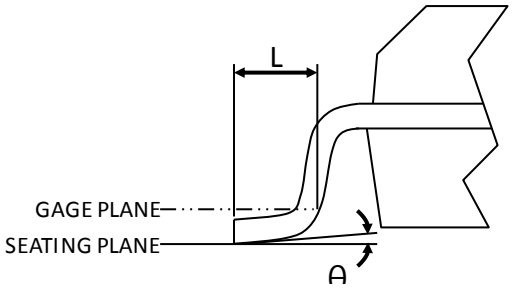



TOP VIEW




SIDE VIEW




DETAIL "A"

| SYMBOLS | Dimension in mm | | | Dimension in inch | | |
|---|---|---|---|---|---|---|
| | MIN. | NOM. | MAX. | MIN. | NOM. | MAX. |
| A | -- | -- | 1.75 | -- | -- | 0.069 |
| A1 | 0.10 | -- | 0.25 | 0.004 | -- | 0.010 |
| B | 0.31 | -- | 0.51 | 0.012 | -- | 0.020 |
| D | 4.90 BSC | | | 0.193 BSC | | |
| E | 3.90 BSC | | | 0.153 BSC | | |
| e | 1.27 BSC | | | 0.050 BSC | | |
| H | 6.00 BSC | | | 0.236 BSC | | |
| L | 0.40 | -- | 1.27 | 0.016 | -- | 0.050 |
| θ | 0° | -- | 8° | 0° | -- | 8° |

Notes：

1. CONTROLLING DIMENSION：mm
2. JEDEC OUTLINE：MS-012 AA

## 28 Appendix: Reference Document

Sonix provides reference document for users to help them quickly familiar SN8F5000 family (downloadable on cooperative website: www.sonix.com.tw).

| Document Name | Description |
|---|---|
| SN8F5000 Starter-Kit User Manual | This documentation introduces SN8F5000 family all Starter-Kit, providing the user selects an appropriate starter-kit for development. |
| SN8F5000 Family Instruction Set | The document details the 8051 instruction set, and a simple example illustrates operation. |
| SN8F5000 Family Instruction Mapping Table | This document supplies the information about mapping assembly instructions from 8-Bit Flash/ OTP Type to 8051 Flash Type. |
| SN8F5000 Packaging Information | This documentation introduces SN8F5000 family microcontrollers' mechanical data, such as height, width and pitch information. |
| SN8F5000 Debug Tool Manual | This document teaches the user to install software Keil C51, and helped create a new project to be developed. |

# SN8F5702 Series Datasheet

## 8051-based Microcontroller

**Corporate Headquarters**
10F-1, No. 36, Taiyuan St.
Chupei City, Hsinchu, Taiwan
TEL: +886-3-5600888
FAX: +886-3-5600889

**Taipei Sales Office**
15F-2, No. 171, Songde Rd.
Taipei City, Taiwan
TEL: +886-2-27591980
FAX: +886-2-27598180
mkt@sonix.com.tw
sales@sonix.com.tw

**Hong Kong Sales Office**
Unit 2603, No. 11, Wo Shing St.
Fo Tan, Hong Kong
TEL: +852-2723-8086
FAX: +852-2723-9179
hk@sonix.com.tw

**Shenzhen Contact Office**
High Tech Industrial Park,
Shenzhen, China
TEL: +86-755-2671-9666
FAX: +86-755-2671-9786
mkt@sonix.com.tw
sales@sonix.com.tw

**USA Office**
TEL: +1-714-3309877
TEL: +1-949-4686539
tlightbody@earthlink.net

**Japan Office**
2F, 4 Chome-8-27 Kudanminami
Chiyoda-ku, Tokyo, Japan
TEL: +81-3-6272-6070
FAX: +81-3-6272-6165
jpsales@sonix.com.tw

**FAE Support via email**
8-bit Microcontroller Products:
sa1fae@sonix.com.tw
All Products: fae@sonix.com.tw